RasPi

DESIGN
BUILD
CODE

9

Get hands-on with your Raspberry Pi

OPERATING SYSTEM

SUPERTEST

Build a SMART ALARM

Welcome

Unless you've got a Jam to go to, mornings can be tough. Get your Raspberry Pi involved, though, and all of a sudden you can freshen up your wake-up and start your day properly with an alarm clock that's personalised to suit your routine. Need a 17-minute snooze? No problem. Only wake up to the sound of singing narwhals? That's fine too. Swipe forward a few pages to crack on. This issue we're also getting you started with servos, now that you've mastered the GPIO ports, as well as casting a critical eye over the biggest distros for your Raspberry Pi to find out which OS is best. And if you're using an older Pi with just a couple of USB ports, you'll love our keyboard and mouse guide.

Gavin Thomas

Deputy Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

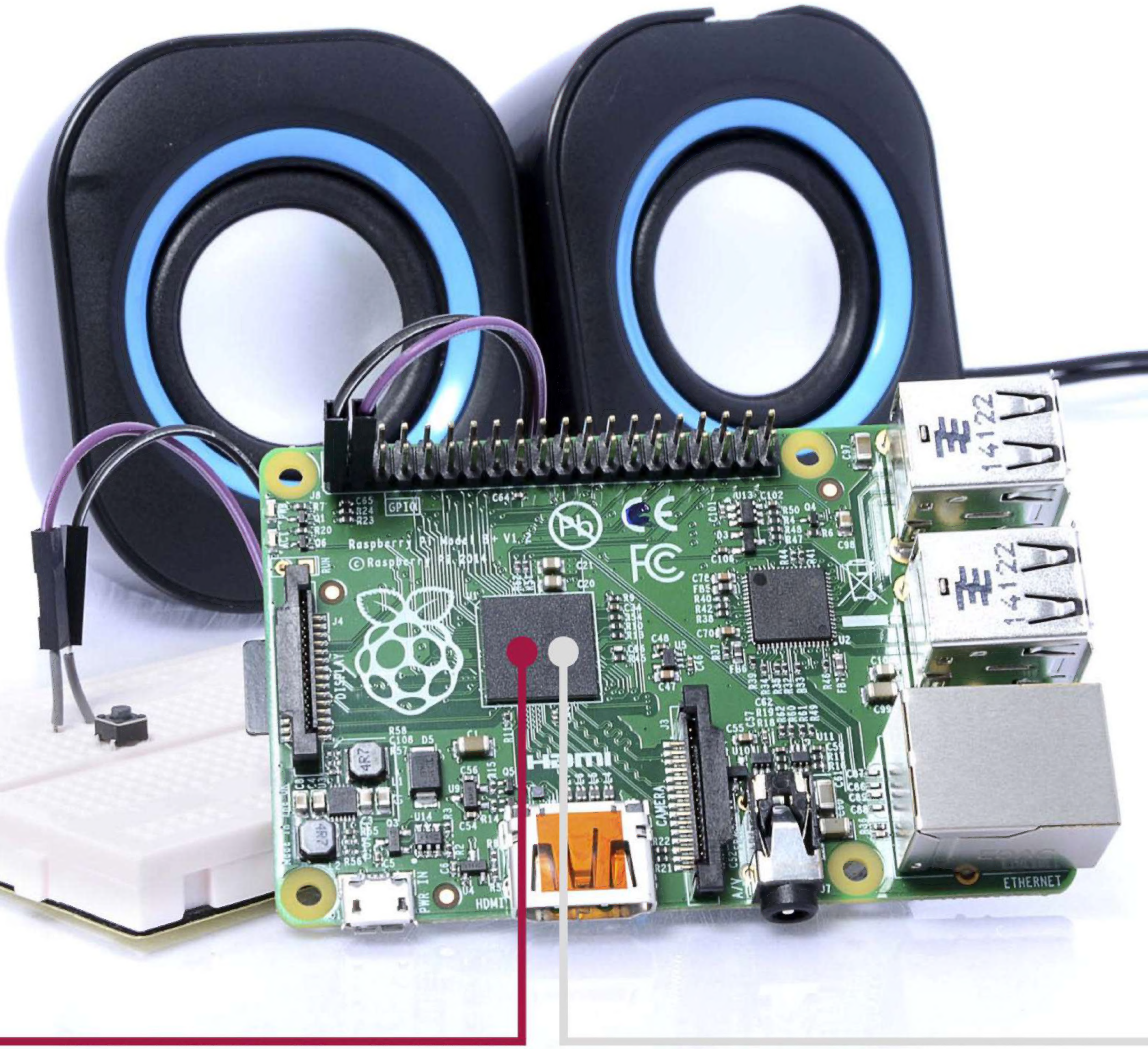
Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk

Contents

Smart alarm clock

Turn your Raspberry Pi into a better alarm clock using Python and a single button



Advancements in alarm clock technology are hardly exciting news reports you'd see on the ten o'clock news but it's something very real in this day and age. Anyone with a smartphone has access to a world of apps tailored to helping users get a better night's sleep, through sleep-cycle sensing and a gradually increasing alarm volume to softly wake them up. If you don't have a smartphone or find the current selection too limited, you can always make your own smart alarm clock using a Raspberry Pi. In this tutorial we'll show you how to program the Pi to play a bit of white noise, use a custom alarm sound and have it wake you gently.

THE PROJECT ESSENTIALS

Speakers or some form of audio output

Push-button switch

01 Update your Pi

To make sure the Raspberry Pi will work as expected, it's best to get it updated first of all. Do this with:

```
sudo apt-get update && sudo apt-get upgrade
```

This will also mean that when installing new software the latest version is installed as well.

02 Install audio playback

For this tutorial, we'll be using VLC to playback the white noise and the alarm sound. It can be used quite easily by Python for the tasks we want it to perform. Install it in the terminal using:

```
$ sudo apt-get install vlc
```

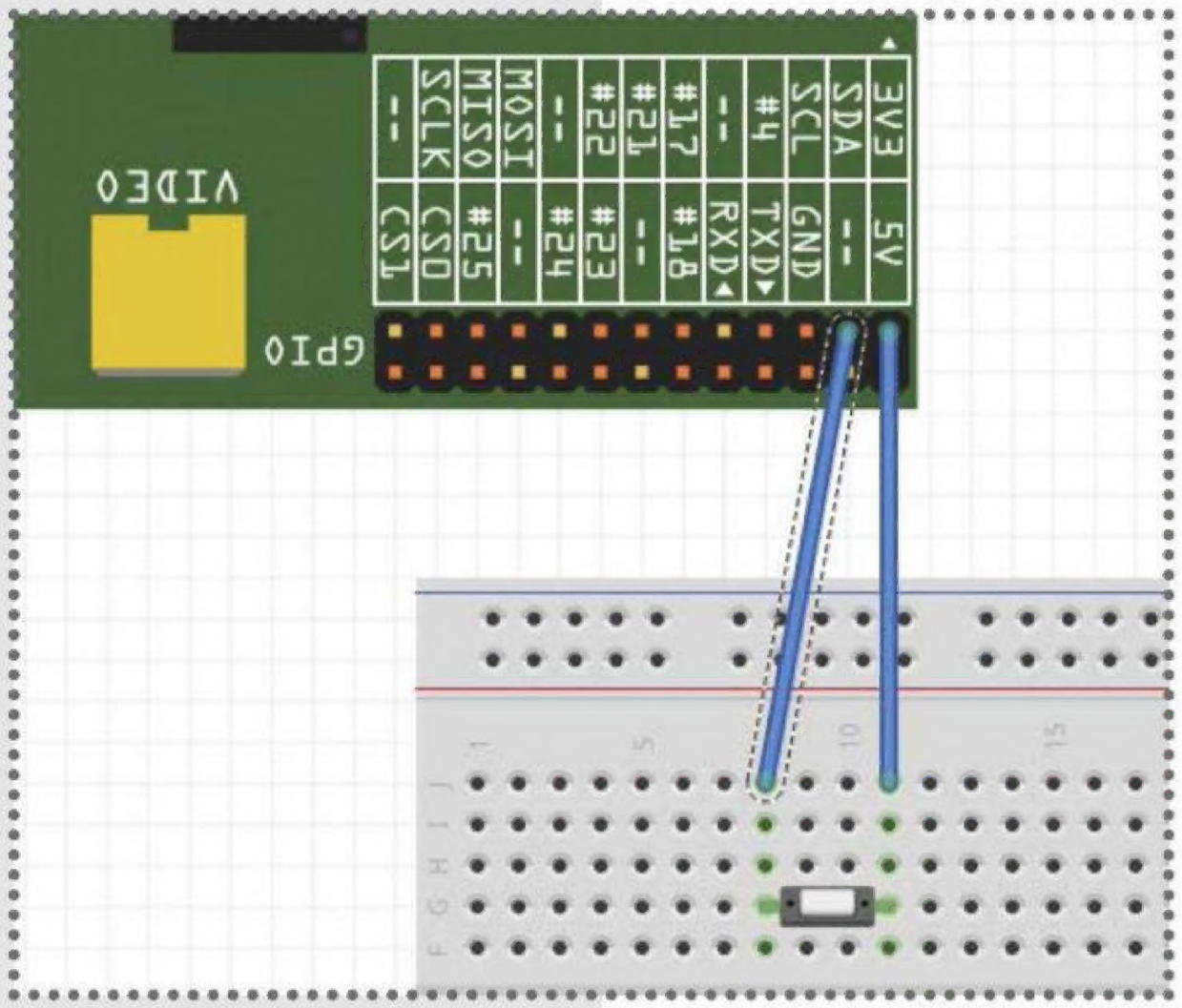
"The alarm clock will also support playlists if you prefer listening to music"

Clock case

Breadboards are excellent for prototyping circuits but if you want to start keeping your Pi by your bed, it might be better to get a case and lose the breadboard. As the wiring is very simple, you could easily solder it up and connect it to the GPIO ports directly. Drilling a hole in a case will keep everything tidy.

03 Wire up your button

Follow along with the Fritzing diagram above – don't worry, the wiring is very simple and simply requires a button and a breadboard. In the current example, we'll only be using one button to shut off the alarm but we'll cover how you could add a snooze button later on.



Above Push buttons are cheap and widely available. Try **adafruit.com/product/367**

04 More Pi preparations

Once the button is wired up, turn the Raspberry Pi back on; it should boot normally if everything is wired up correctly. Once the desktop has loaded, create a new folder in home called alarm. In the version of the code we have created, this is where the necessary mp3s should be held.

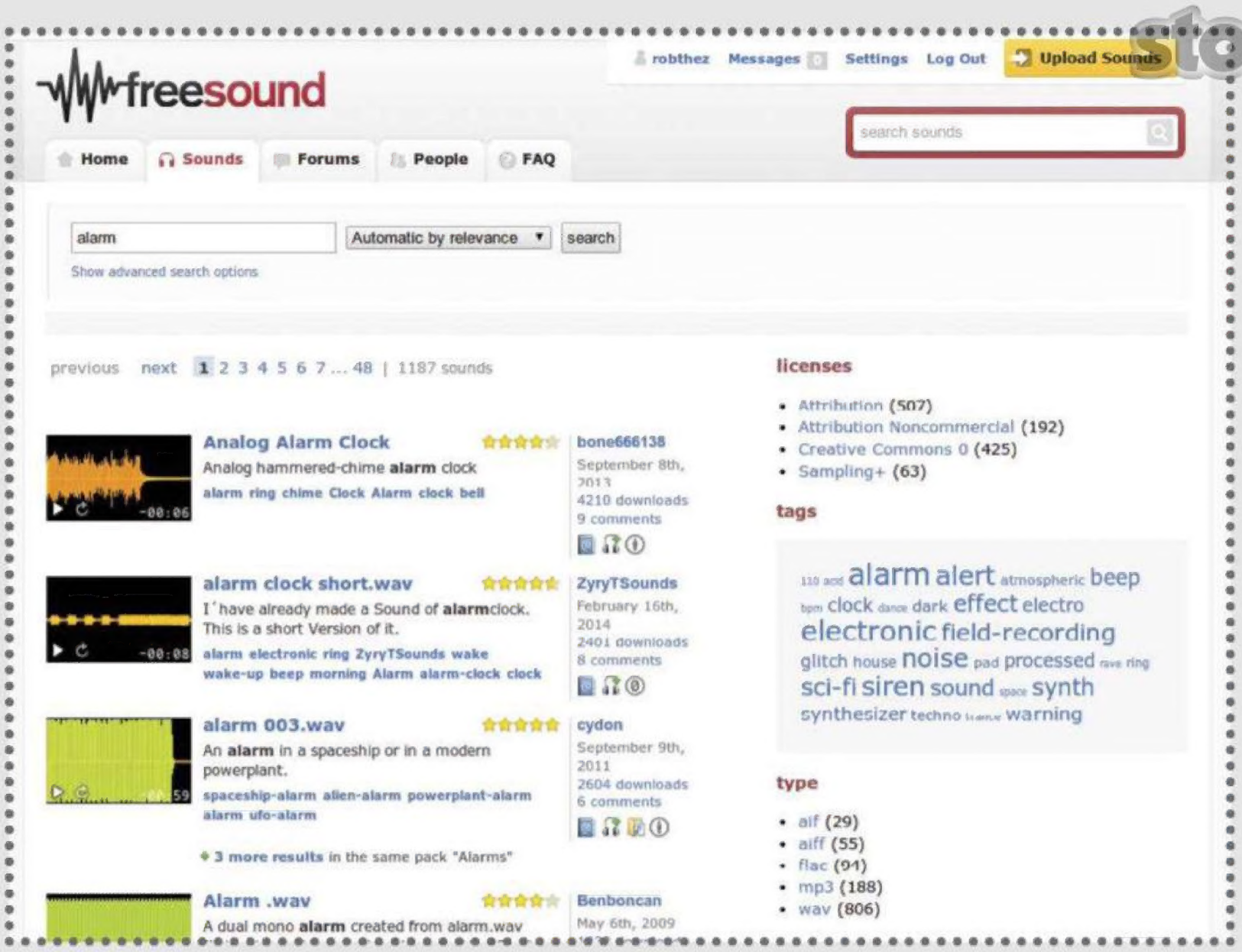
05 Get the code ready

We've got the full code on the next page, broken down into chronological pieces so you know exactly what means what. You can either type it up and edit it in the process or download it directly from the Internet using:

```
$ wget http://diskcontent.imagine-publishing.
co.uk.s3.amazonaws.com/lud/Issue144/pialarm.zip
```

06 Set your alarm

In the alarm code there are two variables: alarm_hour and alarm_minute. These determine when the alarm goes off and operate on a 24-hour input. The default is set for 7:30am, so change it to the time at which you wish to be woken up.



Left Freesound.org is a great source of Creative Commons Licensed audio files

07 Audio choices

You can use whatever you wish for both the white noise and alarm sound. These need to be placed in the directory we created for the alarm clock files and either named the way they are in the code unless you plan to modify that particular part. The alarm clock will also support playlists if you prefer listening to music to fall asleep to.

08 Keep upgrading

While it is a smart alarm clock, there are definitely things that can be done to improve it. Have a look over the code on the next page and give it a test to make sure the volume levels and increments are to your liking, and once you're happy we'll give you some ideas on how to further improve it.

"The code makes sure the current time is the same as the alarm time before activating"

The Code PI ALARM

01 Only basic modules are needed for the code – the time for the alarm, GPIO for the button, and system for the process of calling VLC and changing volume:

```
#!/usr/bin env python
import time
import subprocess
import RPi.GPIO as GPIO
```

02 The basic alarm settings are kept here. The time is in 24 hours for the hour variable, and the volume is a percentage value. Change these to suit your needs:

```
alarm_hour = 7
alarm_minute = 30
alarm_volume = 10
```

03 The white noise file begins to play. It uses the vlcwrapper and is called within a shell. The repeat option is important to make sure it plays all night long:

```
subprocess.call('amixer set PCM 10%', shell=True)
subprocess.Popen('vlc-wrapper /home/pi/alarm/whitenoise.mp3 --repeat -I dummy',
shell=True)
```

04 This sets up the GPIO port to accept an input from the button, which will be used to kill the alarm clock. No snooze function yet, though!

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3 , GPIO.IN)
```

The Code PI ALARM

05 The while loop is created so that the code can keep track of the time, using localtime as part of the process:

```
while(True):
  current_time = list(time.localtime())
  current_hour = current_time[3]
  current_minute = current_time[4]
```

06 Using the if statement, the code makes sure the current time is the same as the alarm time before activating. The sub-processes use the command line to first kill VLC, which stops the white noise, before restarting it to play the alarm again in repeat:

```
  if current_hour == alarm_hour and current_minute == alarm_minute:
    subprocess.call('killall vlc', shell=True)
    subprocess.Popen('vlc-wrapper /home/pi/alarm/alarm.mp3 --repeat -I dummy',
shell=True)
    time.sleep(2.5)
```

07 If the button is pressed, VLC is killed. This simply stops the alarm and you can wake up:

```
    while True:
      if GPIO.input(3)==0:
    subprocess.call('killall vlc', shell=True)
```

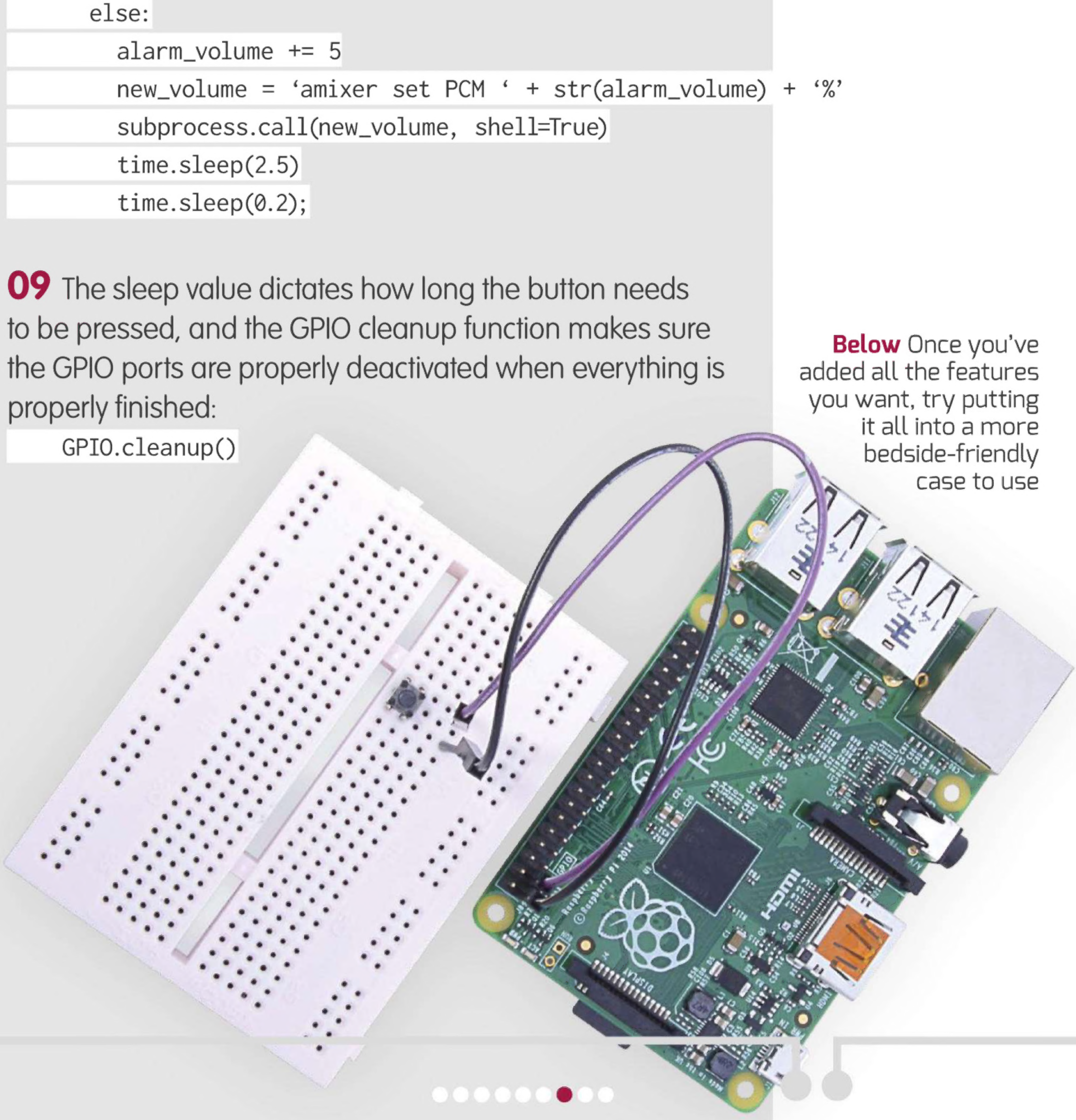
"The XLoBorg from PiBorg is a motion-sensing device. In this case it could be hooked up to record movement during sleep"

08 If the button isn't pressed, the value for alarm volume is increased by five and this value is then passed on to the amixer command line function to increase the volume by 5 per cent. It then waits 2.5 seconds before the volume is increased again:

```
        else:
            alarm_volume += 5
            new_volume = 'amixer set PCM ' + str(alarm_volume) + '%'
            subprocess.call(new_volume, shell=True)
            time.sleep(2.5)
            time.sleep(0.2);
```

09 The sleep value dictates how long the button needs to be pressed, and the GPIO cleanup function makes sure the GPIO ports are properly deactivated when everything is properly finished:

```
        GPIO.cleanup()
```



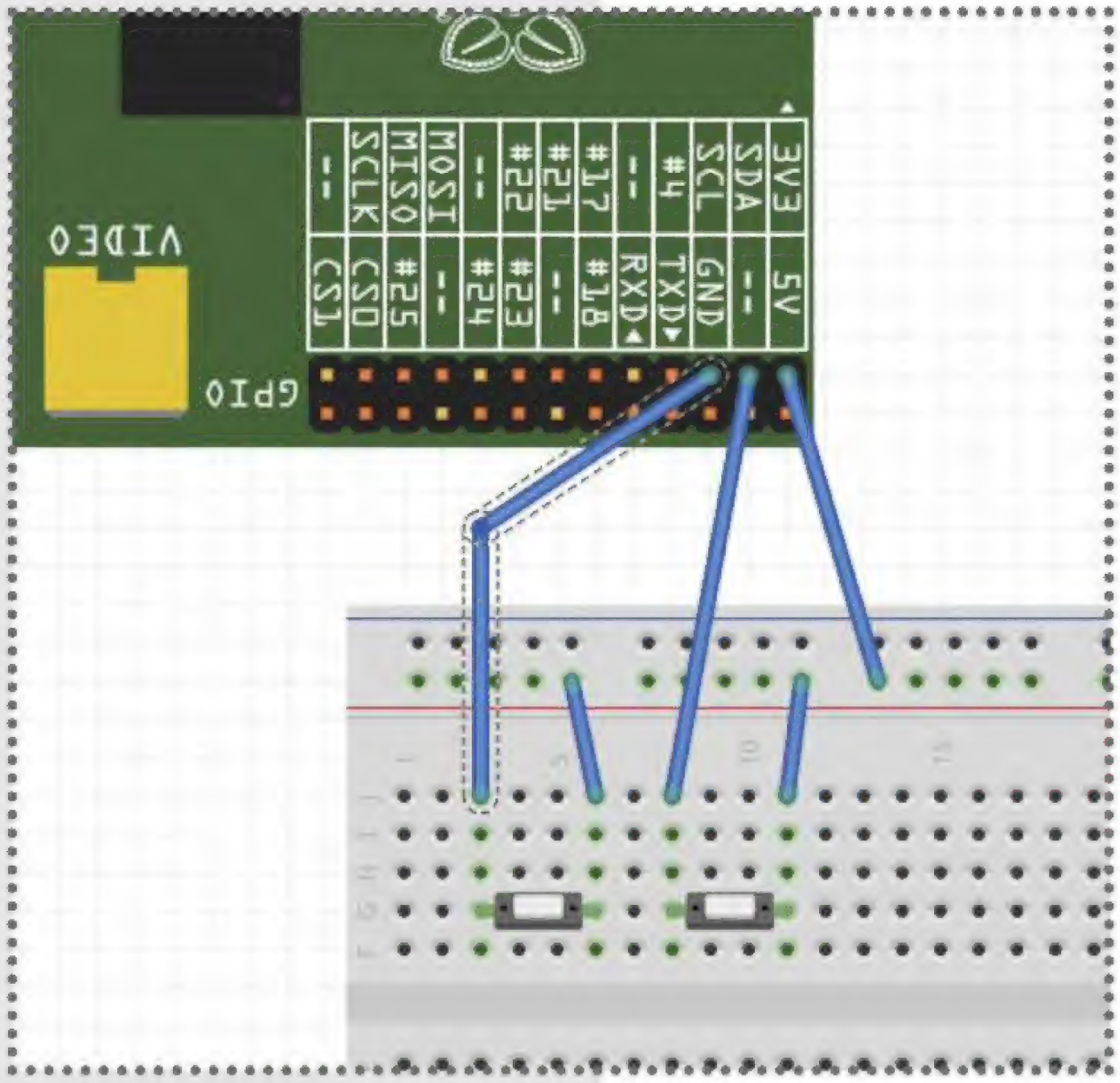
Below Once you've added all the features you want, try putting it all into a more bedside-friendly case to use

Make it smarter

These upgrades can improve the alarm clock to make it more useful and allow you to get rid of your phone as an alarm clock...

Snooze button

The snooze button is a time-honoured tradition of getting just five more minutes in bed. While a lot of movement-based alarms aim to get around this, it's still a feature a lot of us like to use, and we'd be lying if we said it wasn't a vital part of a functional alarm clock as far as we're concerned. As this alarm doesn't have the ability to sense movement (yet), adding a snooze button is perfectly fine.



Above If you're using an older Pi for this, why not solder the connections up?

Adding another push button is fairly simple, just wiring it up as in the Fritzing diagram. The input needs to be set up like the alarm off button and an elif statement can be added to the final while loop to deactivate the alarm and add a few extra minutes to the alarm time. If you want the white noise to reactivate then you'll have to create another subprocess.popen to play the file again for the duration of the snooze.

Motion sensing

The motion sensing functionality on a lot of smartphones is relatively easily implemented as Android phones usually have gyros and magnetometers to register movement. These are very sensitive and with the right code and

analysis they can be used to figure out what sleep cycle you're in, choosing the right time to wake you up.

The Raspberry Pi is a slightly different beast. The XLoBorg from PiBorg is a motion-sensing device that can be wired up to the Pi for a number of functions. In this case it could be hooked up to record movement during sleep, lying beside you on a bed. With the right level of testing and coding you can use it to create a sleep profile, though it requires a bit of trial and error and some research into sleep cycles to understand exactly what's going on.

Screen

There's one flaw in the current alarm clock: there's currently no visible clock when the system is running. It would likely not help your sleep if a monitor is hooked up to the Raspberry Pi all night just to squint at the time in the corner.

The PiTFT is an accessory we looked at in the last issue of **RasPi**: it's roughly the size of the Raspberry Pi and there are cases that allow it to fit snugly on top as one compact Raspberry Pi system. Using Tkinter or Pygame you can create a simple interface that displays the time, or even go further and have it flash when the alarm is going off and display data such as the weather or appointments. The screen also supports touch functionality, so you could make it even more like its smartphone counterparts with snooze/stop buttons on-screen.

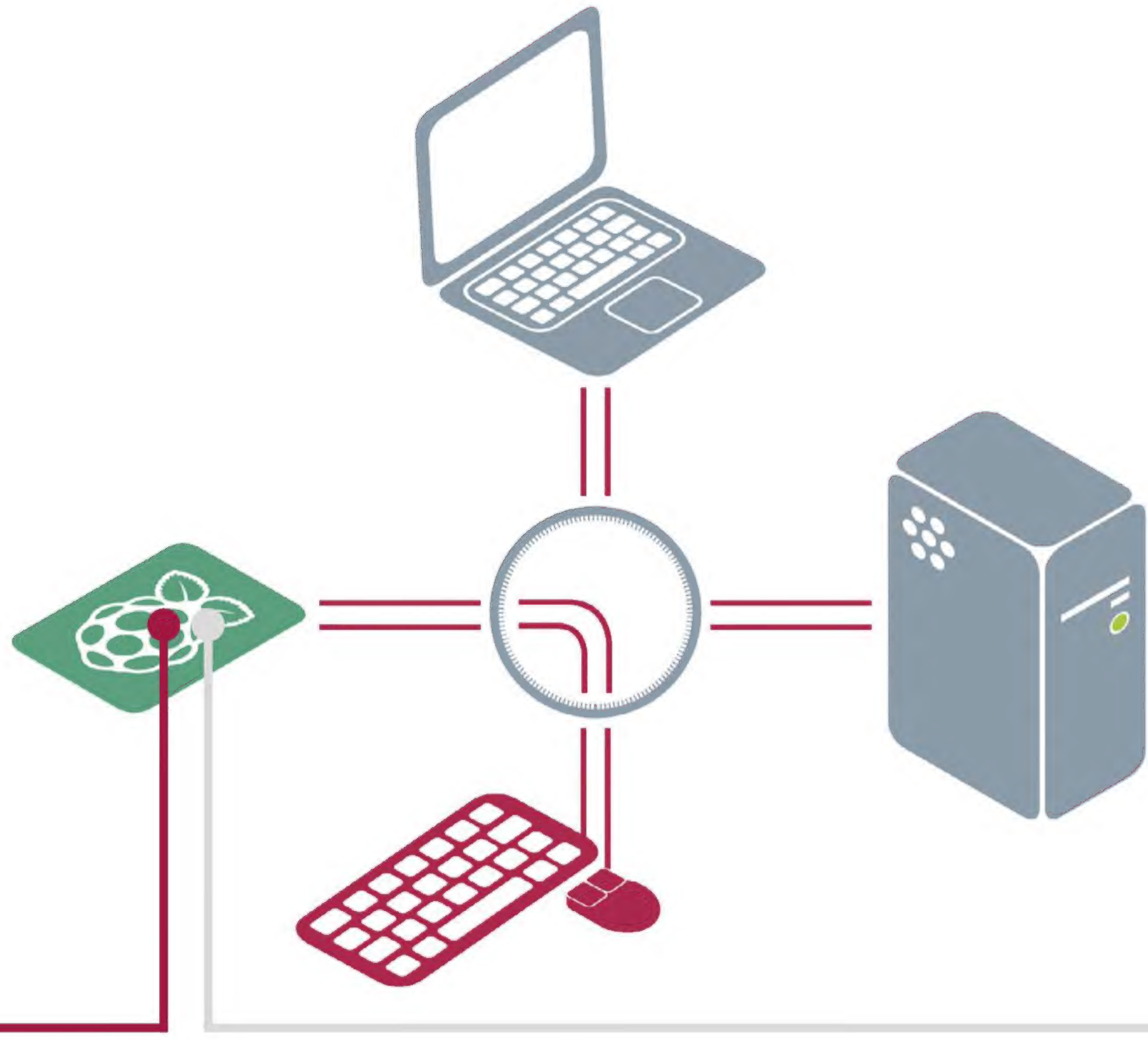
Snooze time

How much time you spend snoozing is up to you. Five and ten minutes are normal on a lot of clocks, but if you're feeling adventurous you could set one for 45 minutes. This gives you time to get some REM sleep if you're still tired – just change your alarm times accordingly!

Below You can pick up the PiTFT for about £35 in most Pi stores: **bit.ly/19w5FqV**

Network and share your keyboard and mouse

Borrow a mouse and keyboard from another PC, using only your Raspberry Pi and Synergy



THE PROJECT ESSENTIALS

Synergy

Host computer

One issue we sometimes find with the Raspberry Pi is the lack of USB ports. We don't always have the luxury of using a powered USB hub, and it can become a bit of a hassle to juggle a mouse and keyboard with other devices. Instead of using a hardware solution for this, you can always try a software solution – one that opens up the uses of the Raspberry Pi as well. The Synergy program lets you share the mouse and keyboard of one system with other systems on the same network, acting as a virtual KVM. In this tutorial we'll learn how to use your main computer's input devices on your Raspberry Pi, as well as how you can look into making it a server.

01 Install Synergy

Synergy is available from the Raspbian repositories. We can install it by using the following:

```
$ sudo apt-get update
$ sudo apt-get install synergy
```

This will go through the basic installation process as normal and Synergy will then be put inside the Accessories folder.

02 Start up Synergy

Start up Synergy on the host computer and choose the 'Server' option for the moment. We'll cover how to use it as a client later, and how to use the Raspberry Pi as a server for the mouse and keyboard.

"Synergy lets you share the mouse and keyboard of one system with other systems on the same network, acting as a virtual KVM"

03 Encryption and passwords

Here, you can set up whether or not the connection is encrypted. This is useful for stopping key loggers from being able to snoop on your information, or random clients from connecting and hijacking your mouse. Provide a password and then click Finish.

Above Select the Server option to use this machine's mouse and keyboard for the Raspberry Pi

04 Server naming

Once the process has finished, go to Configure Server. Your host computer will be put virtually in the centre of your array of displays and you can drag and drop it around, along with any connected screens. Double-click on the server to change its name, making it easier to remember and find. Add a new screen and call it pi.

05 Starting and connecting

Once you're happy with the setup, click Start to be able to accept client connections. To connect from a Raspberry Pi, enter the following:

```
$ synergyc --name pi [IP Address of host]
```

It will be recognised as 'pi' on the host system.

06 Autostart Synergy

To make sure it starts every time you turn on the Pi, we need to create an LXDE autostart file by using the following commands:

```
$ sudo mkdir -p ~/.config/lxsession/LXDE
$ sudo touch ~/.config/lxsession/LXDE/autostart
$ sudo nano ~/.config/lxsession/LXDE/autostart
```

And then add the following to autostart:

```
~/.startsynergy.sh
```

07 Start file

Open and populate the startsynergy file with:

```
$ sudo nano ~/.startsynergy.sh
```

```
#!/bin/bash

killall synergyc
sleep 1
synergyc --name pi [IP address of host]
exit 0
```

08 Permissions

Finally, to finish it off you'll need to run:

```
sudo chmod 777 ~/.startsynergy.sh
```

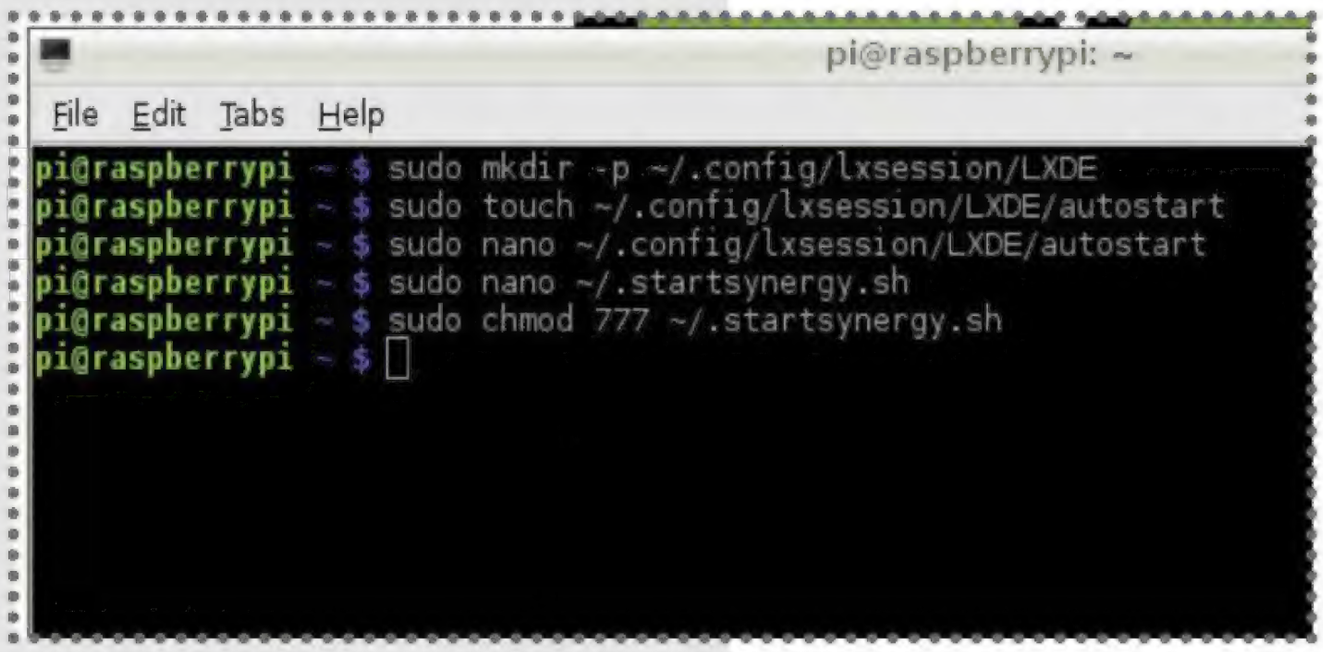
This will autostart, and hopefully autoconnect, Synergy whenever you turn it on. The Raspbian client is a little old, so if you get a problem you may need to compile the latest version from source.

09 Pi server

Setting up the Raspberry Pi as a server is a little more involved and uses the synergys command. It allows you to listen for clients on specific addresses. You then need to create a separate configuration file to arrange the displays – however, you can load one from a different computer and edit it.

"This will autostart, and hopefully autoconnect, Synergy whenever you turn it on"

Below Chmod 777 gives everyone all (ie read, write and execute) permissions for this file



DiceBot

Request a roll from the tweet-powered dice machine, built with Ruby, Python and Pi with an antique game

So where did the idea come from – was it a personal project?

I worked on it myself. I got a little help on the website from some others at Intridea, but we've been working as a company just exploring various interfaces and social machines – Internet of Things, things like that. So it's kind of how this idea came about. We have a few projects that we're working on at the moment that are in a similar vein but this is the first one we've published. So I came across this little dice roller and I thought 'Hey, this would make a perfect Internet-controlled device'. And it would be a fun project, using something old and retro.

The dice roller looks amazing – where did you manage to find that thing?

I actually found it on eBay and supposedly it's from the Twenties – it shows a lot of age and wear. It's a little tin device and it used to have a tiny push button on the side with a little gear. You push that in and, as it was spring-loaded, you'd let go and it'd pop, and the little platform would spin with a couple of little dice in there.

So did you rig up some motors with the Raspberry Pi and get the original mechanism working then?

I pulled out the old mechanism and replaced it because it wasn't very reliable, since it was quite old. So yeah, I connected a motor to it and connected that to a little motor controller, a servo controller, and that to the Raspberry Pi. Then, on the Raspberry Pi, I ran some software to control everything.

Dave Naffis, co-founder of Intridea, is an entrepreneur and software developer. He's contributed to a number of open source projects and also ran a RoR consultancy

Below Dave replaced the old push-to-spin mechanism with a small motor

So which method are you using to read the tweet feed, then?

I'm using a Ruby script that I wrote and I'm using a library called Tweetstream. It's a Ruby library, so it's basically listening on Twitter for any mention of @IntrideaDiceBot as well as the hashtag #RolltheDice. Every time it runs across that it queues a job, putting it into something called Firebase. From there there's another script that's running that listens for jobs on that queue, and whenever it finds one it picks it up and it does the rest. It spins the device through the GPIO pins on the Raspberry Pi, and that's using a Python script. So, it calls the Python script to spin the device through the GPIO, and then it takes a picture using the raspistill command. Then it uploads that to Amazon AWS, to Amazon S3, and then it runs an OpenCV program I created that basically takes that image and runs a Canny on it, trying to figure out how many pips are visible. From there it takes the image, takes the results of the OpenCV output, and then it tweets back the result as well as prints the results back into Firebase, so the website can display it in real-time.

How long did it take to make this?

It was actually pretty straightforward; it was just a weekend project to build the device and to write the scripts. The scripts are pretty straightforward and simple as well. I also had to build the platform that it sits on. It used to just be a bunch of loose parts – you know, the Raspberry Pi, the controller and the dice spinner. It's kind of hard to set the camera to get the right angle of image over it, so I just built this little platform out of wood around it. It makes it quite portable, and I was able to put the camera into a fixed position so that it's always getting the same angle and the same shot. That seemed

If you like

Interested in wireless networking and Raspberry Pis? Check out the Onion Pi project from Adafruit, where you can build a secure router:
learn.adafruit.com/onion-pi

Further reading

Visit Dave Hunt's blog to learn more about the PiPhone, with information on how to build it and links to the code:
bit.ly/1jNLemR

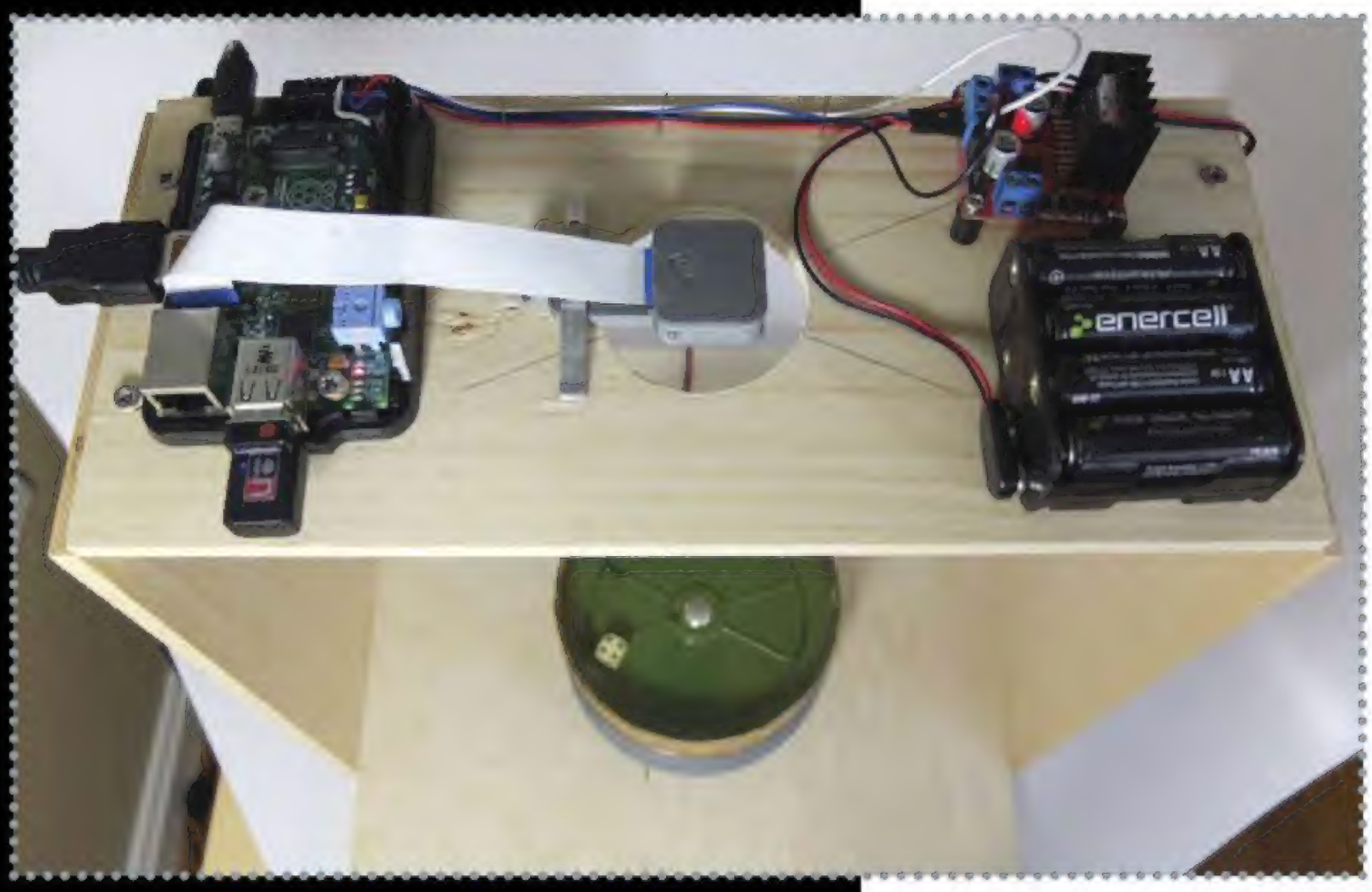
to work quite well. So really it was just a weekend, and then probably another day or two for final assembly and polish, things like that.

Do you have any plans to make it interactive, so that people could play against the tweet feed?

Yeah we've had all sorts of thoughts about stuff like that, and the fun things we could do running and creating some simple games you could play over Twitter. We've thought about using different kinds of dice and doing image recognition for the number versus just the pips on the device, for when you get into higher-sided dice. So yeah, I think there are a lot of possibilities and we're kind of thinking about those – and hopefully soon we can build some fun things around it. Another option is to create a simple API that people can use to get random numbers – so a physical random number generator that they could call via an API from another program or something.

...There – it's spinning. I don't know if you heard that but it's driving me nuts, because I'm sitting next to it!

Below Keep an eye out for a DiceBot upgrade – next on the agenda is reading numbers as opposed to pips, so you can #rollthedice for saving throws...



Raspberry Pi Operating Systems

Most people stick with Raspbian, but is that the best OS for your Pi? Find out in this 12-page super test!

Pidora

The Raspberry Pi Fedora Remix has come a long way since its initial launch, where it lost ground to Raspbian due to its sluggish, buggy behaviour. Pidora 2014, built for ARMv6 architecture and featuring packages from Fedora 20, is impressive at first glance – but are the changes only skin deep?

Download: **bit.ly/1yqRTQM**

RISC OS Pi

A legacy of days gone by, RISC OS Pi is a version of the classic RISC OS for your Raspberry Pi. Developed by Acorn Computers for use on the ARM chipset that they also designed, RISC OS was forked after Acorn Computers was broken up in 1998 and it's a continuation of this project that lives on in RISC OS Pi.

Download: **bit.ly/1vbremd**

Arch Linux ARM

For the ultimate in resource-light, custom-built operating systems, nothing beats an Arch setup. The barrier to entry is steep, but if you're looking to learn the inner workings of Linux systems then few others will give you quite the same opportunity.

Download: **bit.ly/1yqS2Uj**

Raspbian

The Pi community's distro of choice (with a little help from the Raspberry Pi Foundation), Raspbian is popular for a reason. We cast a critical eye over this Debian-based OS to see just what the real advantages and disadvantages are with this distros.

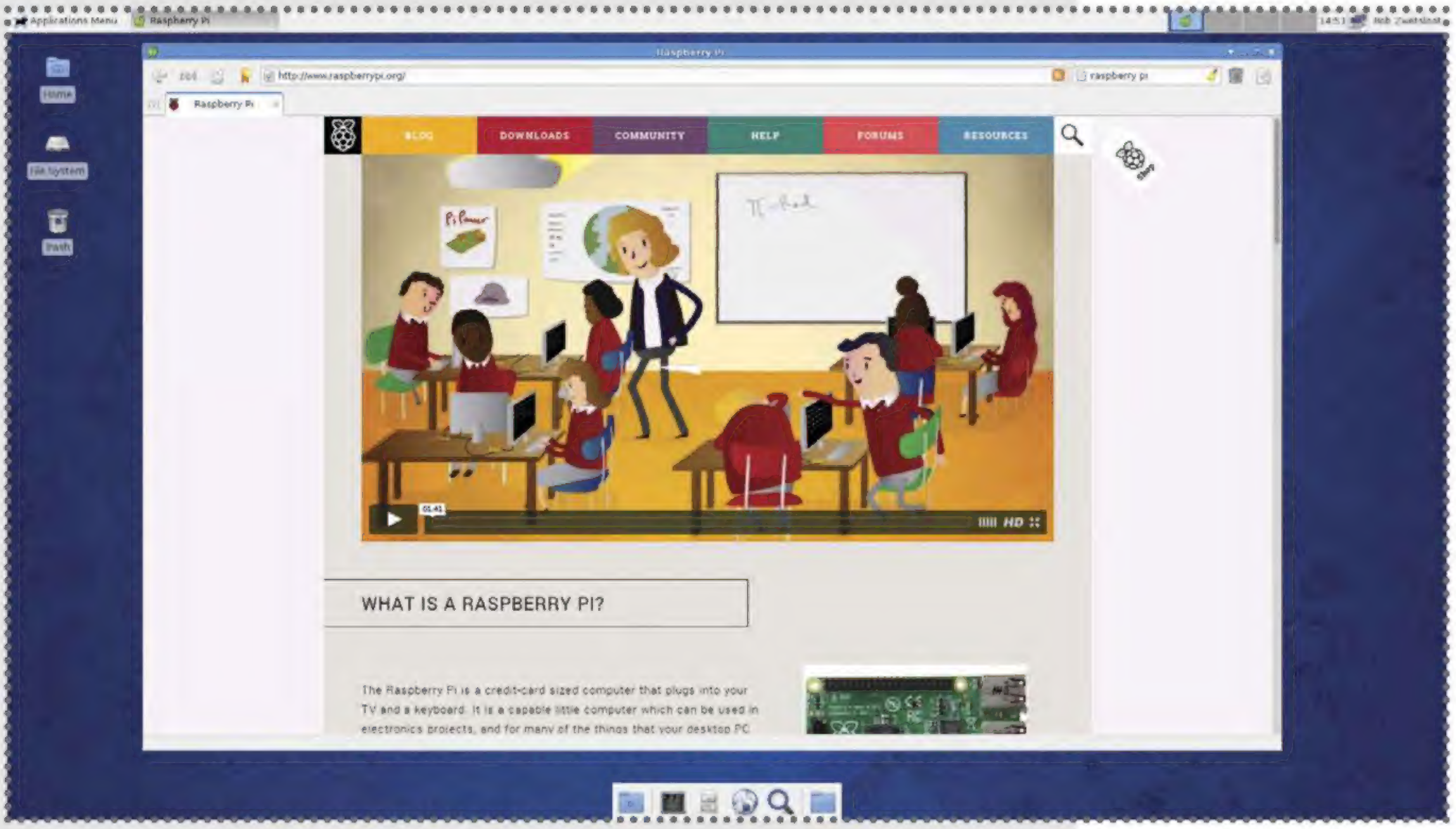
Download: **bit.ly/1spHsdp**

Pidora

There was a spin of Fedora for the Raspberry Pi during the early days of its release, but that was quickly dropped in favour of Raspbian when it proved to be a bit slow and buggy. It was almost two years after this incident that a proper version of Fedora was released on the Raspberry Pi in the form of Pidora. An almost straight-up port of the codebase to the specific ARM architecture of the Pi, Pidora has had a few tweaks to let it run on Pi hardware without much loss in performance, at least.

The very first thing to note is that the problems of Fedora on the Pi in the past are long gone. This is a very mature operating system that is stable and generally runs well on the Raspberry Pi. In terms of speed, it's not as fast as Raspbian or Arch Linux, and in the case of Raspbian this may be due to a number of factors. Pidora uses Xfce, for one, which is known to be a little heavier than the LXDE that

Below Pidora is more mature than the version of Fedora on the Pi from yesteryear



Left Other operating systems are faster but at least Pidora is stable enough

Raspbian uses. Fedora also uses much newer software that is somewhat designed to be run on computers which are slightly more high-end, while Raspbian is based on an older version of Debian with more lenient software. It isn't the biggest difference but it all adds up with the other problems.

Booting from NOOBS is painfully slow and prone to errors – something that doesn't occur with Raspbian or the other distros on NOOBS. Installing straight from the image removes this problem but, unfortunately, Pidora has one other disadvantage over Raspbian: lack of software.

More software is being added all the time but a lot of basic or useful yet obscure packages available for Raspbian just aren't in the repository. The essentials are there and there are plenty of programming choices, but combine this with a lack of official Pi accessory support and it becomes less attractive for project work.

It's definitely not a bad distro, though. If it had ended up being the default when the Pi launched then we'd likely be saying similar things about Raspbian. However, there's very little reason to use it over its Raspbian counterpart.

"A lot of basic or useful yet obscure packages just aren't in the repository"

Overall score: 6

We do really like Pidora but for everything you'd want to use a Raspberry Pi for, Raspbian or Arch is just better

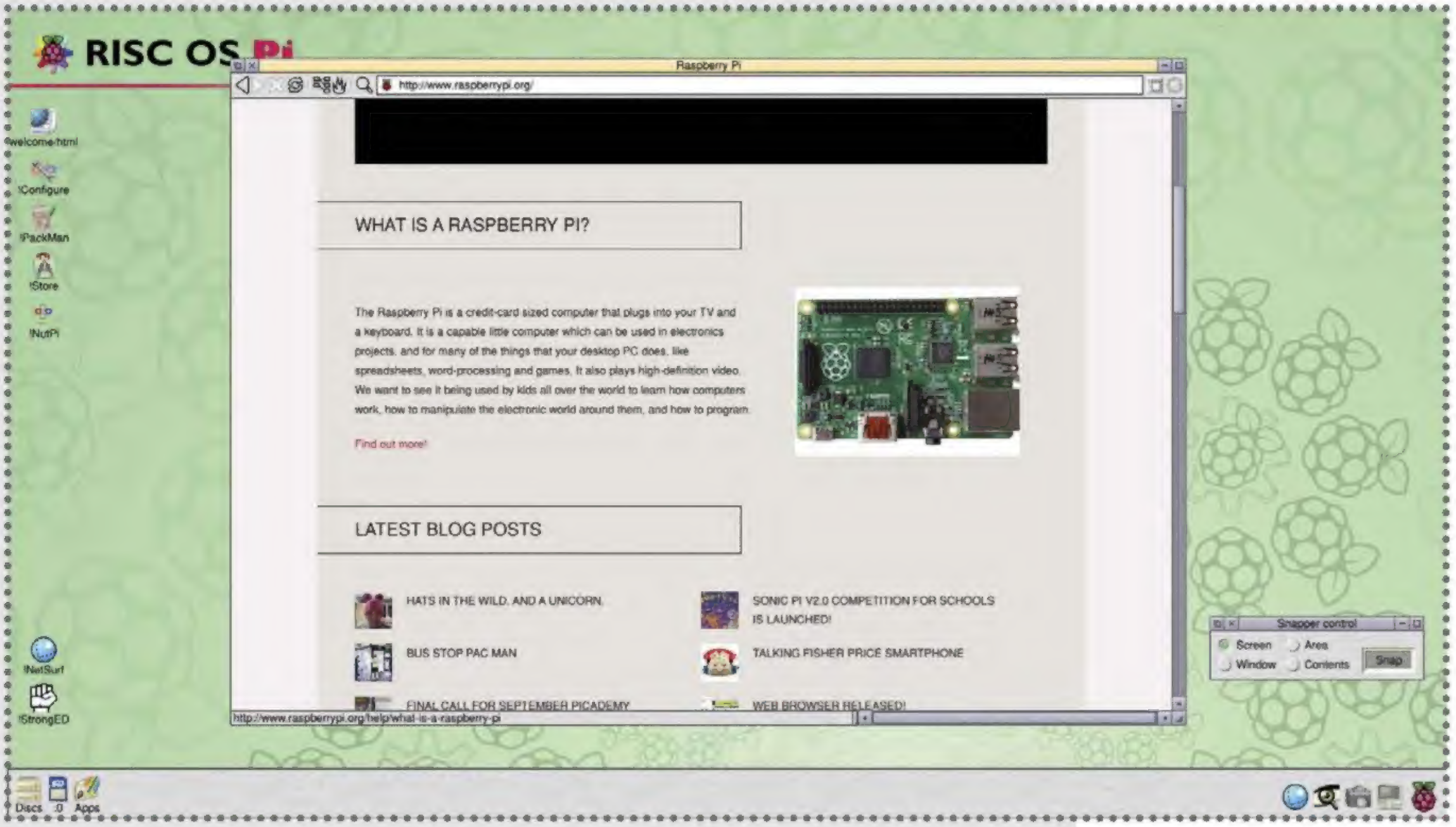
RISC OS

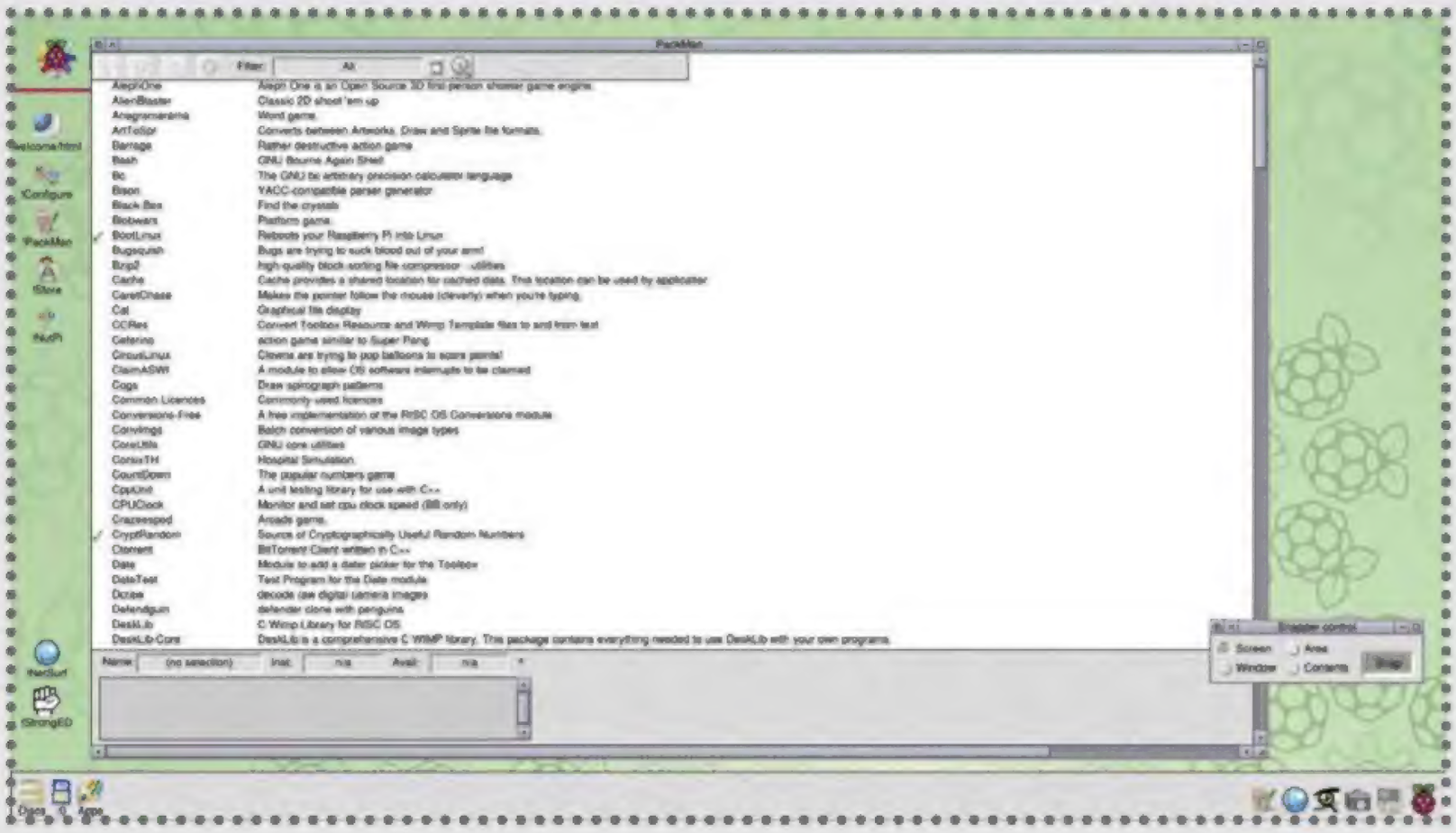
The last time we did a Pi OS group test we slightly enjoyed the blast from the past that was RISC OS, remembering our time in school using Acorn Computers that our Tesco Computers for Schools vouchers paid for. We came away realising that the tint on our rose glasses was a bit strong and struggling to get any screenshots off the SD card.

Since then, RISC OS has been updated a few times, increasing the amount of available software, improving the performance of the OS and finally letting us easily export PNG screenshots for this article. It's still very much the same RISC OS though, a curious novelty from another time that doesn't really fit in with the ecosystems of modern operating systems.

While RISC OS is not Linux it does have a similar file structure and uses a package manager, but otherwise it is completely its own thing. Using RISC is different from

Below Diving into RISC OS is like time travelling back to three decades ago





other operating systems as it uses a three-button system for clicking and opening menus, unlike the two-button standard of today. This can be confusing for new users, along with the way apps open on the desktop and the way they don't always require double-clicks to launch. When it is in a fullscreen window, it covers the panel and a whole host of other quirks. If you're used to the traditional desktop metaphor there are a lot of new workflow processes to get your head around for something you're unlikely to use as your main computer.

As for teaching tools there are very few. You won't really be learning to code on here unless you like using BASIC or already have a bit of knowledge in Perl or C. You can access the GPIO port but it's not very well suited for all projects; Raspbian and Arch would be better.

Things have changed in the 18 months since we first checked out RISC OS on Pi, but it's either not enough or it's just never going to go in the direction of Raspbian. It's definitely interesting, though, so if you have a few hours to play with it then we recommend giving it a shot.

Left There are a lot of different systems and processes to get your head around

"While RISC OS is not Linux, it does have a similar file structure"

Overall score: 4

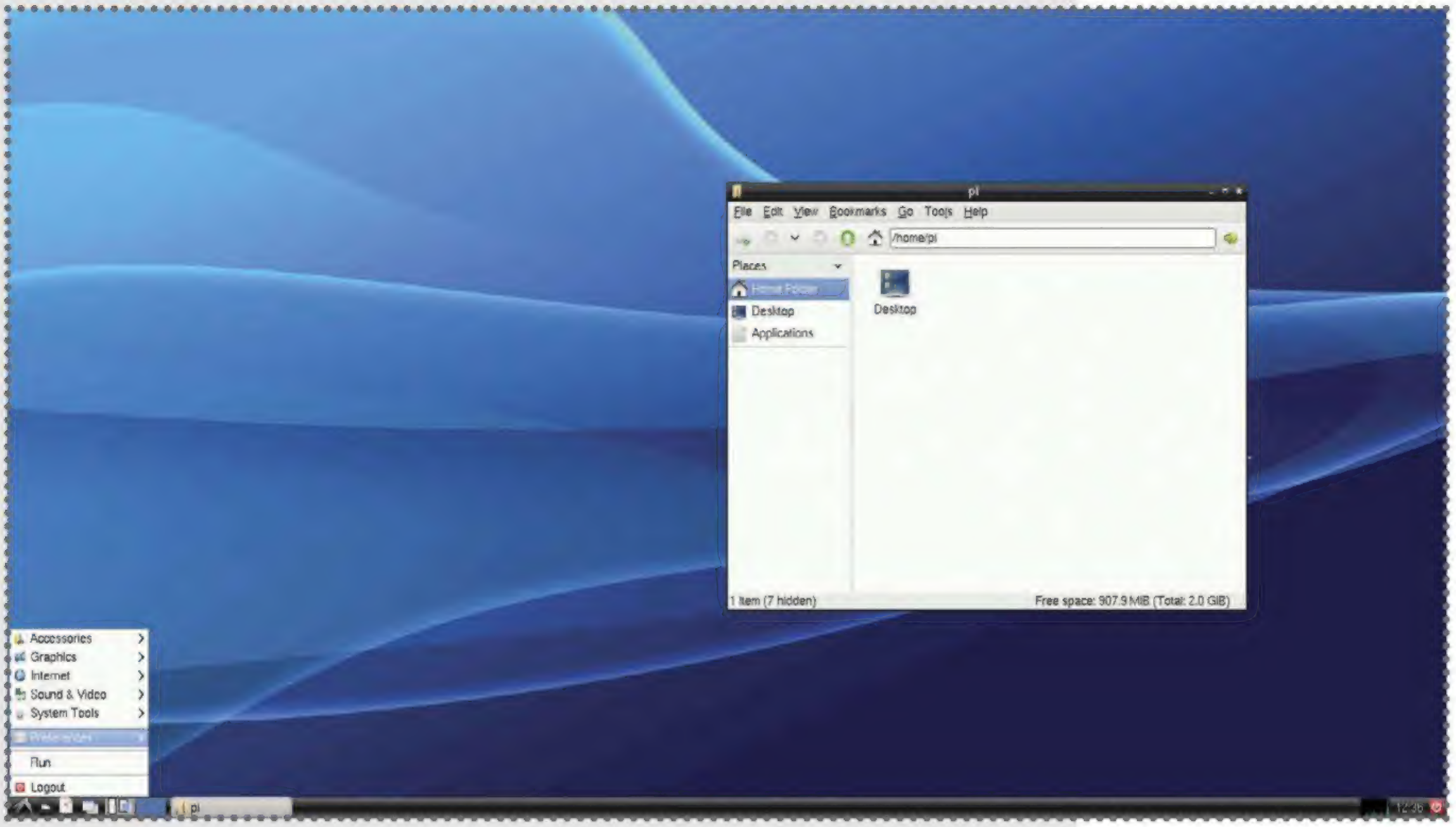
RISC OS is a very different operating system to the others on this test, which isn't a bad thing but it is not conducive to education

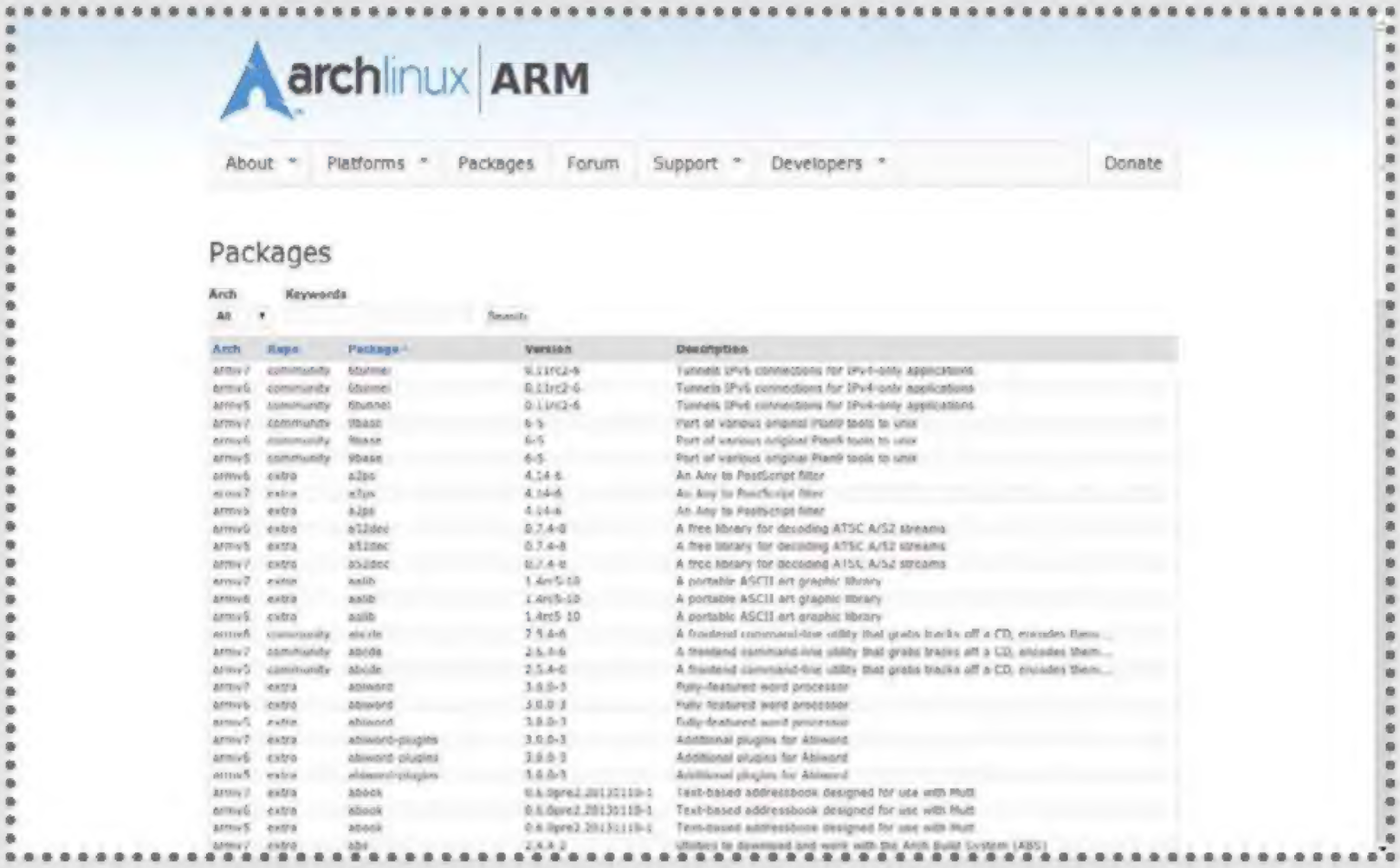
Arch Linux

For just about as long as there's been a Raspbian distro on the Raspberry Pi Foundation's site there's also been an Arch alternative for those a little more adventurous in their choice of operating system. Whereas Raspbian gets you going straight away with an excellent selection of apps, Arch provides users with a command line interface and only a bare smattering of extra packages over the Linux kernel to make it work.

Arch has the bare essentials available to be built upon for a perfectly customised distro. To that end, this version comes with the pacman command line package manager to help you install any available software from the repository including desktops, media players and others. It enables you to have no bloat on the system and the correct setup will always be faster than a modified Raspbian for a lot of projects or applications.

Below Arch Linux isn't clunky; it has the bare necessities available for full customisation





As for Ras Pi-specific functions, it does quite well in this regard too. The Pi camera board is supported and so are the GPIO pins – the latter allowing access to a huge number of Pi accessories, including Adafruit screens and sensors and other physical devices. Some of it can be trickier to use, as you can't rely on pre-built Raspbian software, but it is doable with some of the more complex additions.

For normal users, though, it can be quite terrifying. All knowledge is assumed and, while you can easily find some quick tutorials to get started, it can be quite daunting. Even adding the Pi camera software takes a lot more effort than enabling it in the config menu. It does offer a unique opportunity to learn the intricacies of Linux, at least – which does keep in line with the Raspberry Pi education ethos – but there are a lot of one-off commands that might get lost in the process of getting it from zero to desktop.

For extremely custom projects you may well be better off using Arch over Raspbian. However, for a lot of users there won't be much of a difference in performance.

Left Be careful with one-off commands as they can easily get lost in the many processes

"While you can easily find quick tutorials to get started, it can be daunting"

Overall score: 7

Arch is excellent in its own way but it's definitely not for everyone and really not the best distro for something like the Raspberry Pi

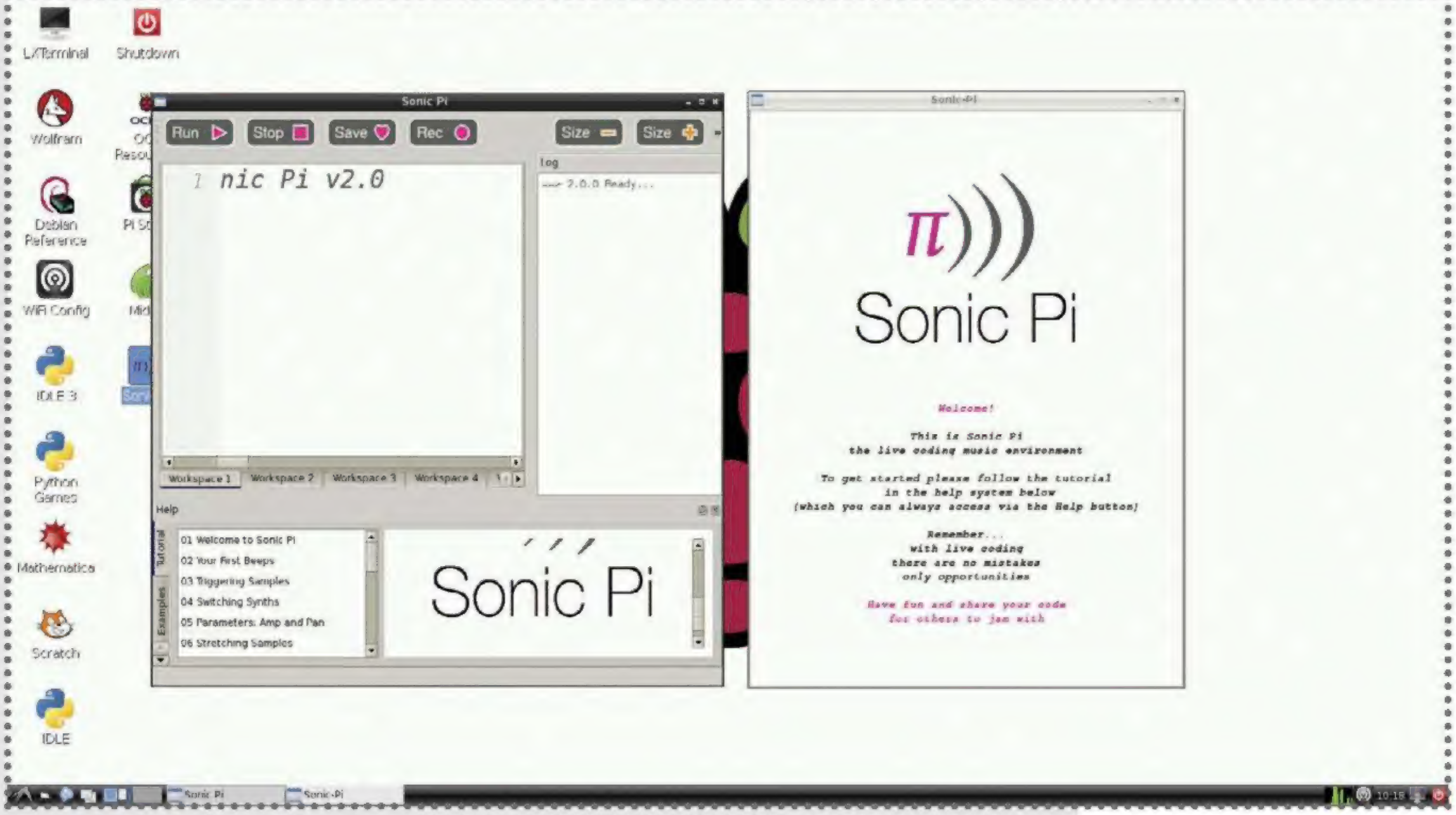
Raspbian

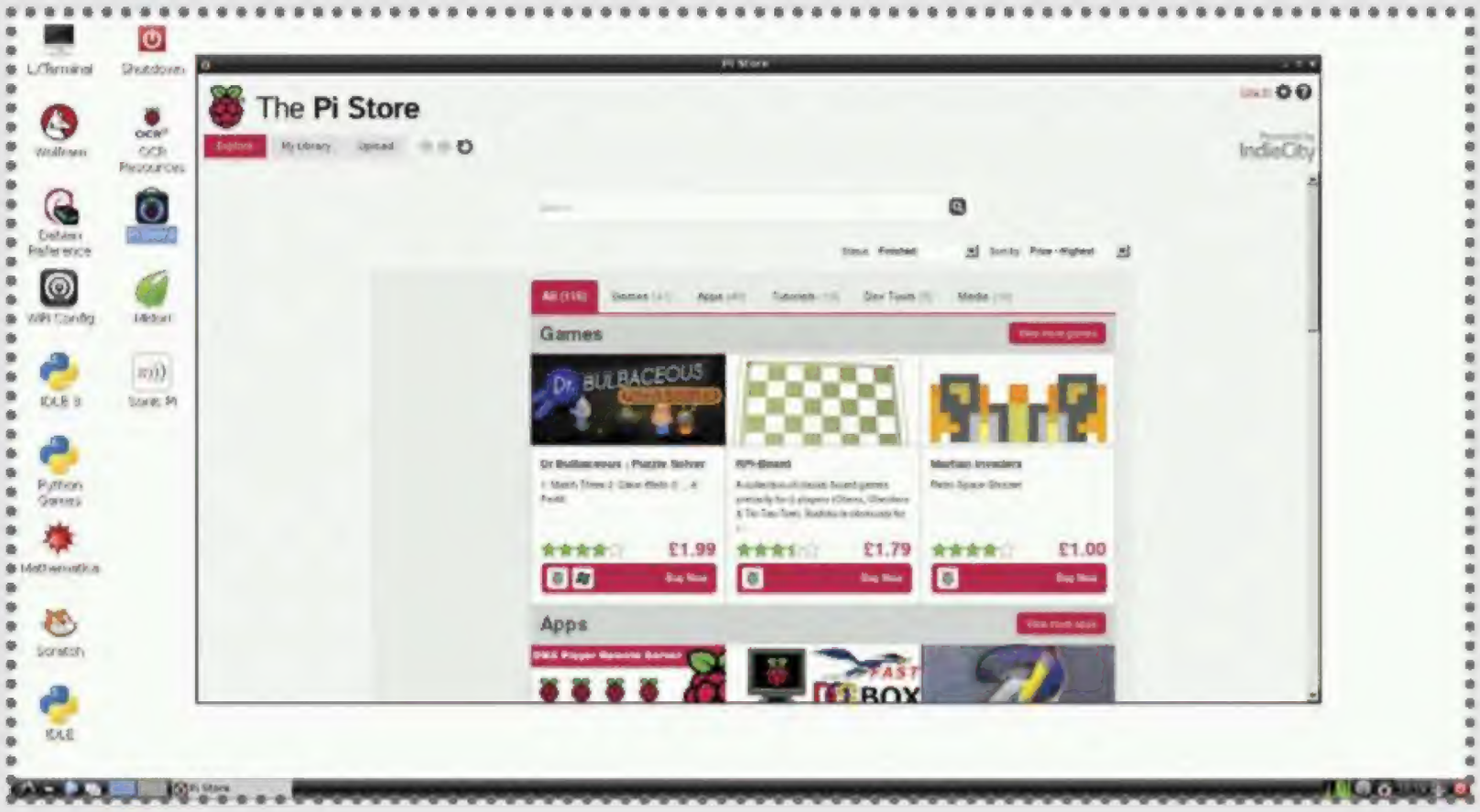
There's no secret as to why Raspbian is the recommended Raspberry Pi distro: it's obviously an easy-to-use distro that can be set up relatively quickly. Thanks to the support of the Raspberry Pi Foundation, it gets a lot more development and attention from the community than any other distro for the Raspberry Pi.

The Debian base of the distro is in no small part responsible for this. It's easy to maintain, and if you ever need to delve into the terminal then the commands are simple and quick enough to use. There are also plenty of software choices through the repositories, which include the majority of the apps you'd use on your main computer; whether or not they're useable on the limited resources of the Pi is another matter.

The major advantage of Raspbian over the other distros is the selection of educational and teaching material

Below Raspbian is the only place you'll be able to use Sonic Pi





included on the distro. From simple software such as Scratch and Sonic Pi that teach coding fundamentals through to a full-on OCR computing course, there are a lot of apps on here that you don't get anywhere else. It also supports all the recommended overclocking limits, the Raspberry Pi camera and any future official hardware add-ons as well.

A benefit of using Raspbian is that a lot of tutorials, projects and third-party hardware run off or are based on it as a standard. It makes it easier to learn coding if that's your goal, or even to replicate these projects if you're not particularly confident in your skills.

Otherwise the distro is quite fast and light. The browser was updated last year to a custom piece of Epiphany-based software that is noticeably faster than the previous Midori effort. Raspian's other custom pieces of software are great at improving its performance but are unique to Raspbian, such as the command line OXM player for playing video and audio.

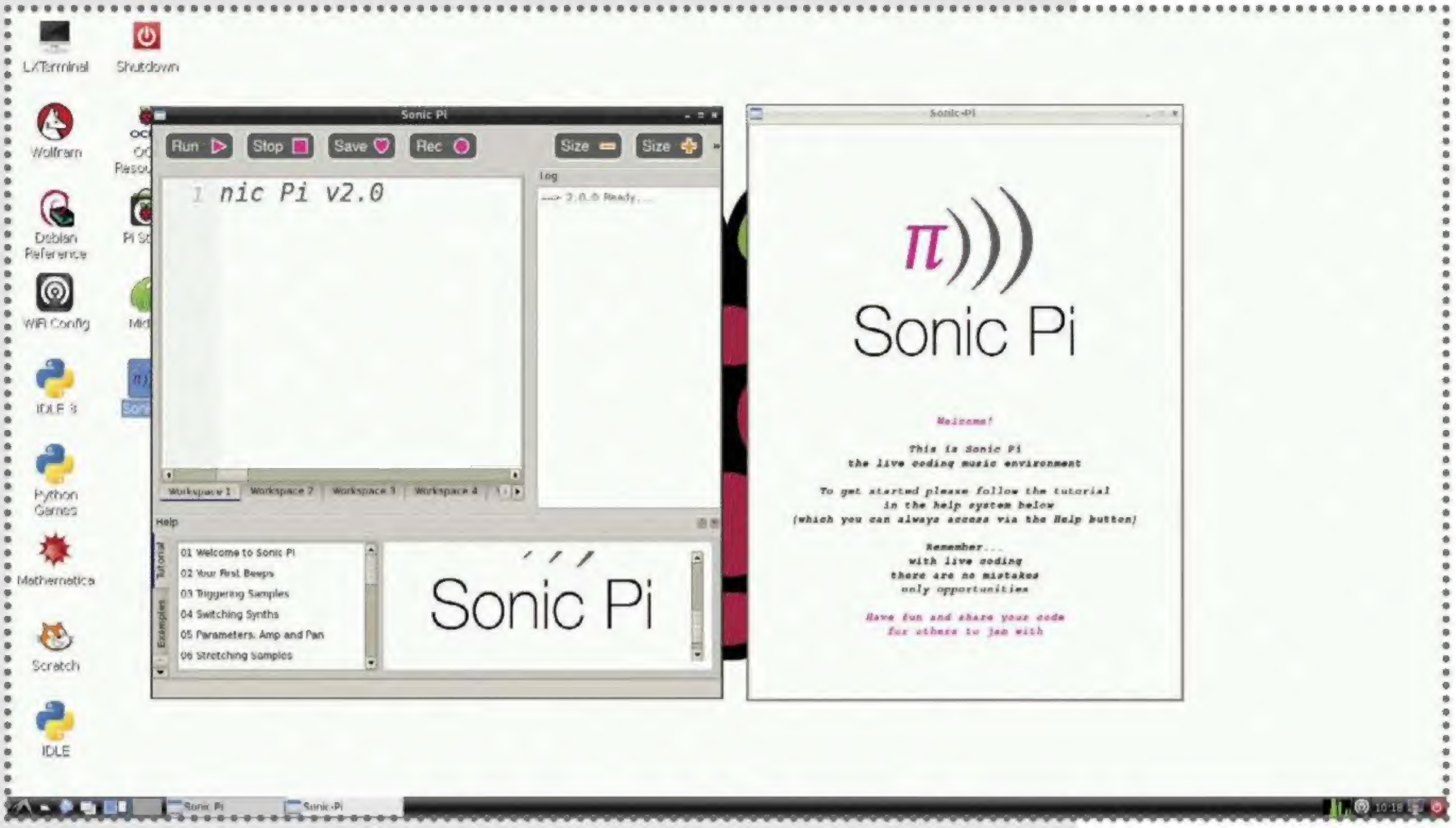
It is essentially the default Linux distribution and this status has given Raspbian a lot of advantages over everything else while still being as flexible as Linux can be.

Left There is plenty of material and software available for Raspbian

"There are also plenty of software choices through the repositories"

Overall score: 9

It's an excellent distro which is designed around all the Raspberry Pi's strengths



And the winner is...

Raspbian

Pidora's strong performance raised an interesting point about these operating systems, and that's the degree to which they're supported by the Pi community. Having won the lead a few years ago, Raspbian has since enjoyed not only the full support of the Raspberry Pi Foundation but also the benefits of documentation – the majority of the tutorials you'll have found for the Pi use Raspbian and, because of this, its leading position has only become stronger. However, looking at Pidora 2014 on its own merits and ignoring the Raspbian bias, it does have some key usability advantages over Raspbian. Not quite enough to make it better, but enough to make us think that this is going to be a real contender in the future.

An honourable mention goes to RISC OS, not only for its preservation of a classic OS enjoyed by many Linux users

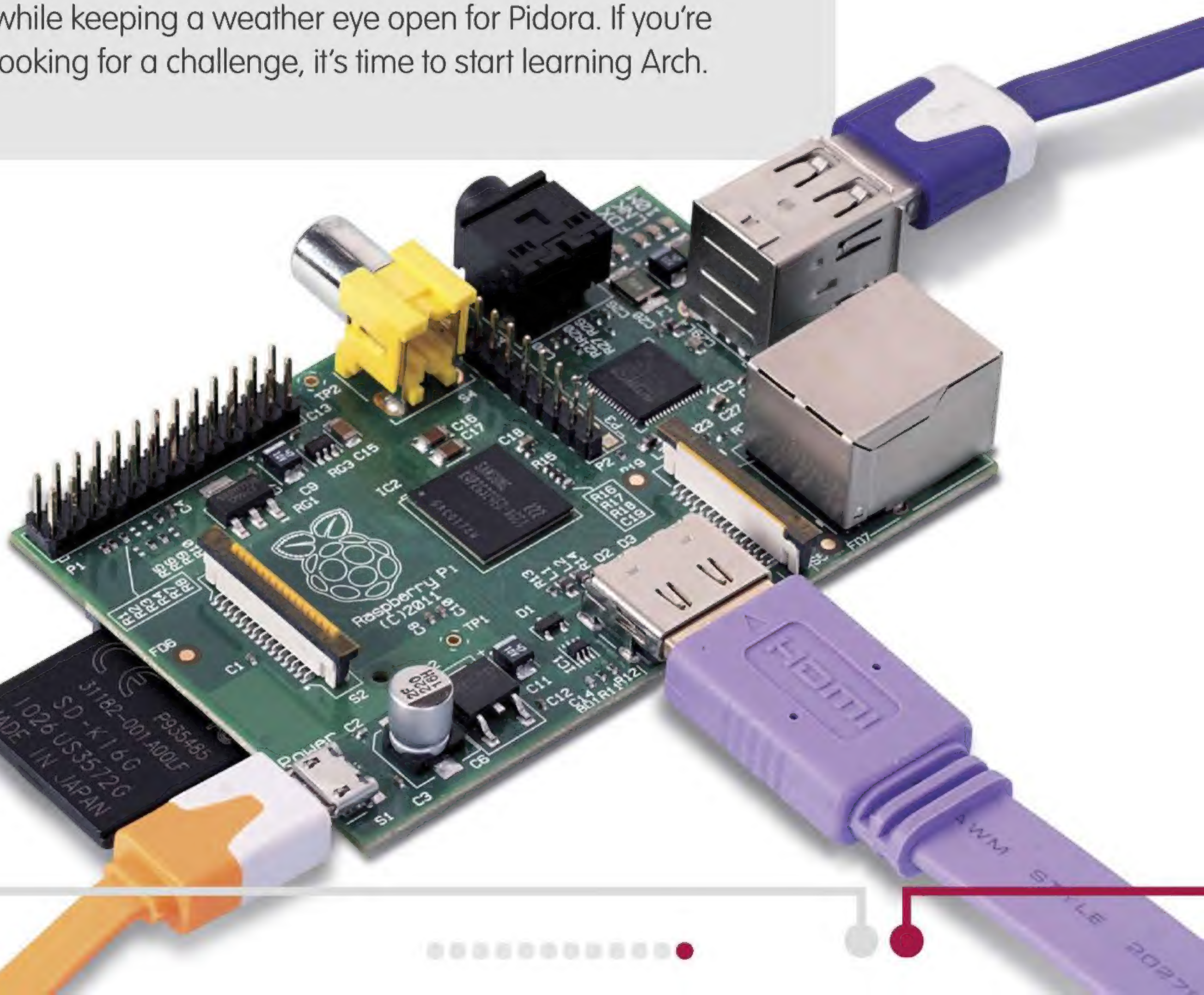
Above A host of tools designed to teach kids to code are built into Raspbian

a few decades ago, but also for providing a completely different kind of desktop experience and helping young Pi enthusiasts discover more about the history of computing. It isn't, unfortunately, going to become our go-to Pi distro, but it's certainly good fun.

As for Arch, whether or not you choose to use this distro will really come down to your previous experience with it on Linux systems. If you know what you're doing – or are prepared to dedicate a substantial number of hours to learning how – then Arch can be the best choice for big, ambitious projects. The absolute control you have over the way you build Arch empowers you to really push your Pi with highly custom setups.

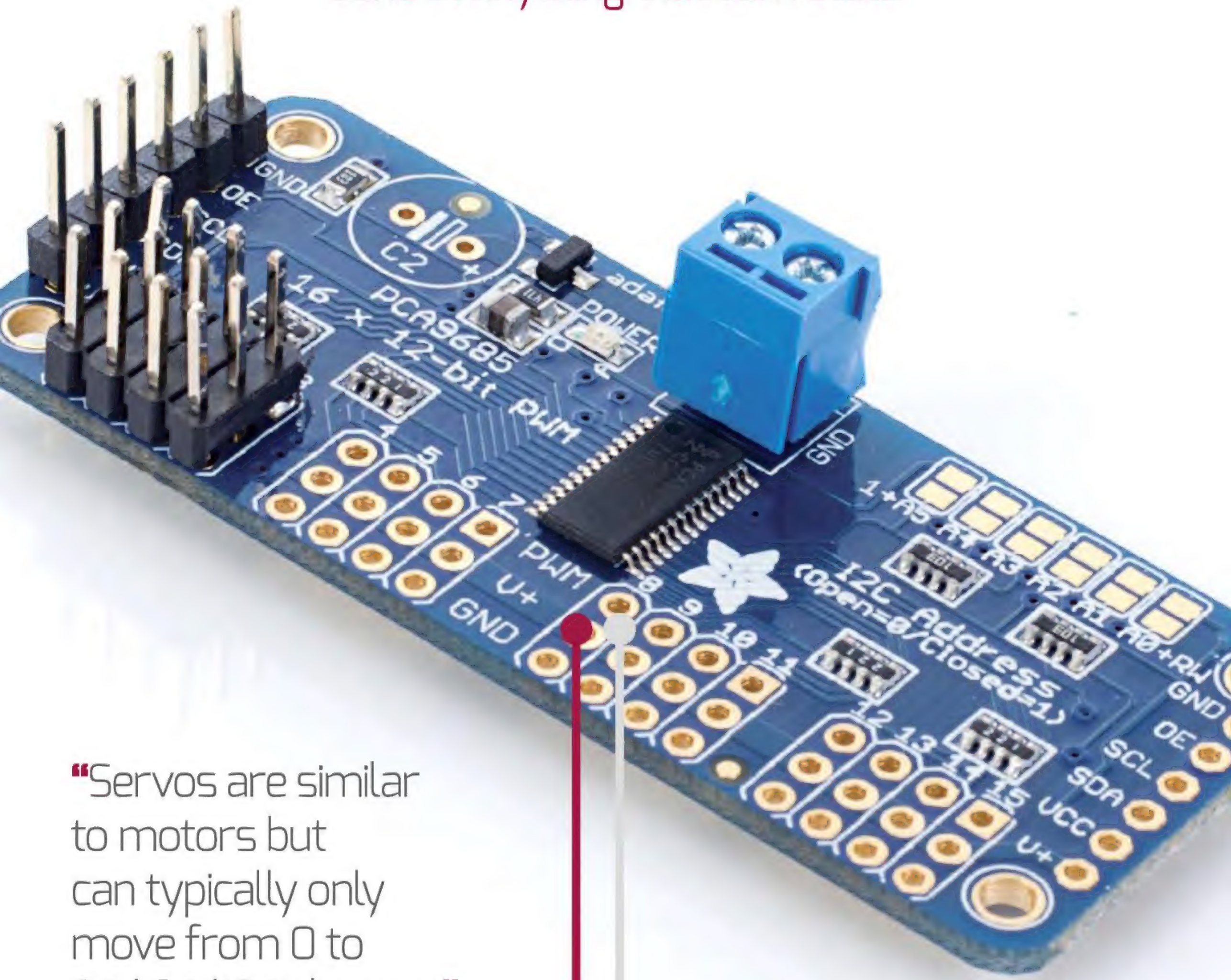
For now, we recommend that you stick to Raspbian, while keeping a weather eye open for Pidora. If you're looking for a challenge, it's time to start learning Arch.

"Having won the lead a few years ago, Raspbian has since enjoyed not only the full support of the Raspberry Pi Foundation but also the benefits of documentation"



Set up and control servos

Use your Raspberry Pi to move parts of a robot or control anything that can rotate



"Servos are similar to motors but can typically only move from 0 to 60/120/180 degrees"

Servos are similar to motors but can typically only move from 0 to 60/120/180 degrees, rather than rotate continuously. Unlike a motor, you send a servo a signal that makes it go to a specific position (eg 30 degrees), making them ideal for applications such as moving parts of robots, or controlling surfaces on an aircraft. That's a little complicated to cover in this tutorial – a simpler use case is using a servo to turn an analogue potentiometer.

Here we are going to create some simple software that will allow a Raspberry Pi to control the volume knob on a simple speaker amplifier, which could be used as part of a remote control project. This step-by-step tutorial assumes that you have already soldered the pin headers onto the Adafruit board. Ready? Let's get going.

THE PROJECT ESSENTIALS

Latest Raspbian image
raspberrypi.org/downloads

Internet connection

Adafruit PCA9685 servo controller
adafruit.com/product/815

5V power supply

Female-to-female jumper cables

A servo suited to your needs

01 Install software and configure I2C

We will start off with the usual 'apt-get' lines. Simply enter the following commands in order to download and install the required software:

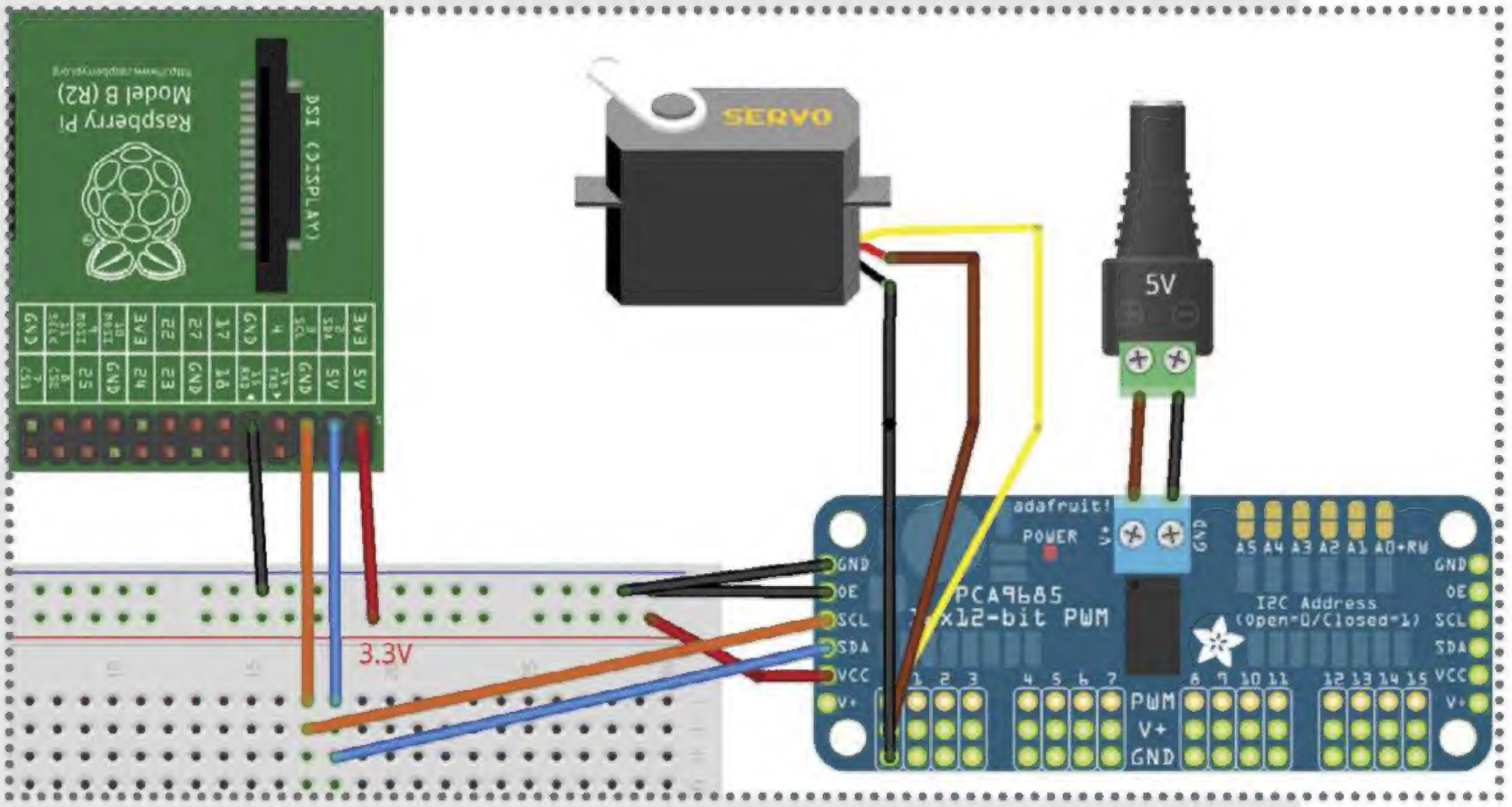
```
sudo apt-get update
sudo apt-get install python-smbus i2c-tools git
git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

By default, the I2C modules are blacklisted. To change this, run:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

...and comment out the following lines by putting a # in front of each of them:

```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```



Above Ensure you get the voltages right to avoid damaging your servo or your Pi

Save changes with Ctrl+O followed by Enter, then exit with Ctrl+X. Then, you also need to add the following two lines to /etc/modules:

```
i2c-dev
i2c-bcm2708
```

Now run `sudo poweroff` to shut down your Pi.

02 Wire the servo controller

Wire up the servo controller, as shown in the circuit diagram. It's essential that all power is disconnected during the wiring process. You also need to make sure that your Raspberry Pi is oriented the correct way. The OE pin on the Adafruit board stands for Output Enable and is active low, which means the output is enabled when the signal is 0V. This is why we have connected it to ground. The VCC pin on the Adafruit board should be connected to 3.3V from the Raspberry Pi and the servo power in the middle of the board should be 5V. The Raspberry Pi's power needs to be stable – and servos often cause a voltage drop – which is why a second power supply is necessary.

```
pi@raspbian ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                 -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- UU -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

Above If you can't see output like this, swipe back a page and check your wiring

03 Test the connection to the servo controller

We will use a program called i2cdetect to detect the servo controller. If you are using an older Raspberry Pi then the I2C bus is 0, otherwise the I2C bus is 1. Type `sudo i2cdetect -y 1`, where 1 is the I2C bus number. You should see an output similar to the one in the image above. Try the other I2C bus and, if not, then it's likely you've wired something wrong.

04 Servo Hello World

Change directory into where you git-cloned the Adafruit Raspberry Pi code (it was probably /home/pi). Then run the following commands:

```
cd Adafruit-Raspberry-Pi-Python-Code
cd Adafruit_PWM_Servo_Driver
sudo python2 Servo_Example.py
```

If your servo is connected to channel 0 of the Adafruit board, it should start moving between its min and max positions.

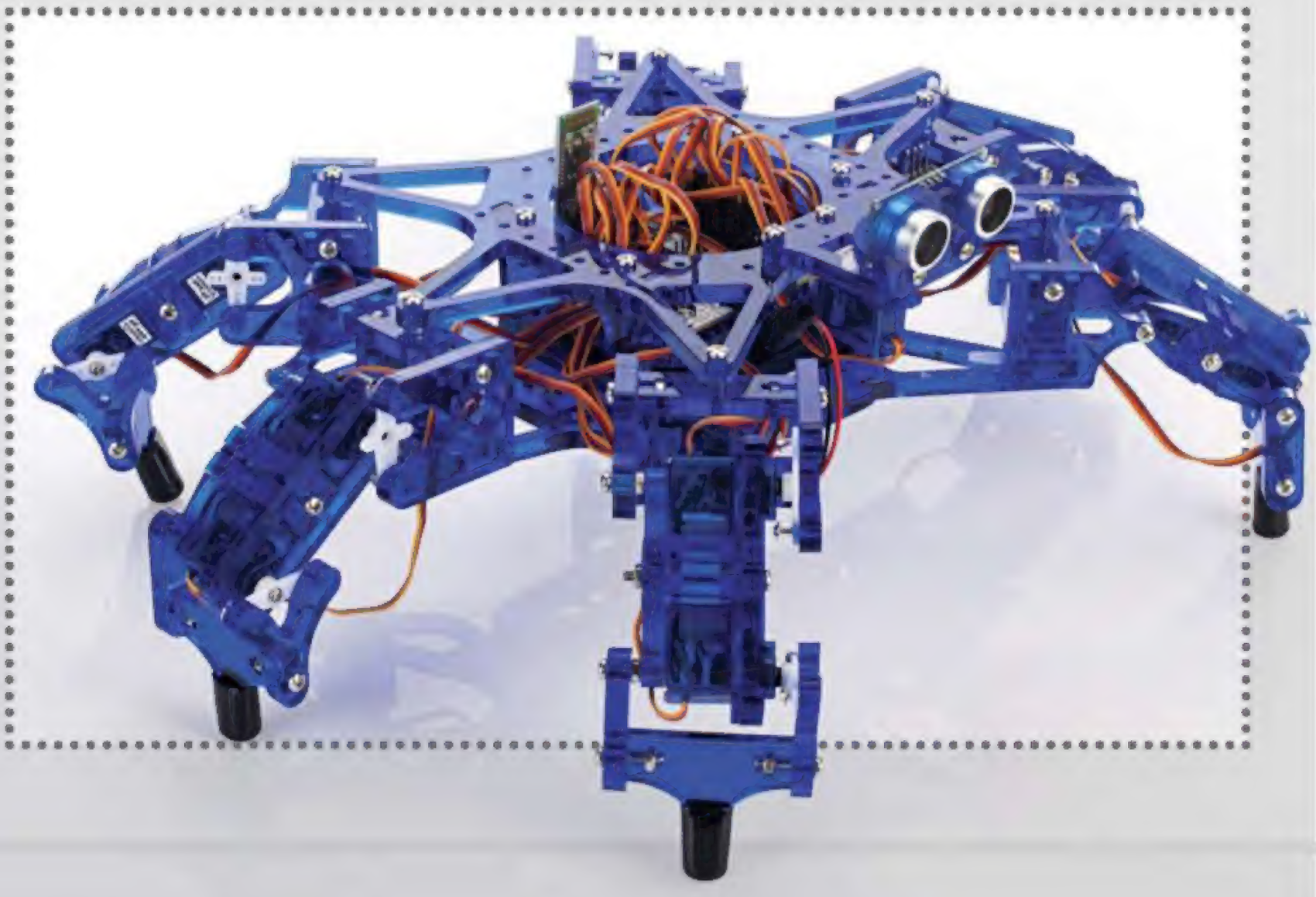
"We will use a program called i2cdetect to detect the servo controller"

05 How a servo works

If you open up the Servo_Example.py code in nano (get it from **bit.ly/18Yayl9**), you'll see that there isn't much to it. This is because the Adafruit board does all of the hard work and the Raspberry Pi just tells it what to do. The Adafruit board is required because a servo requires very accurate pulse-width modulation signals to set its position. This is very hard to do on a Pi because there are so many other things going on. If you look at the two oscilloscope traces, you can see what the Adafruit board sends to the servo. The difference in the duty cycle (the percentage of time that a signal is high) between the minimum position and maximum position is roughly 10 per cent.

06 Run the code

Place the code in the same directory as the Adafruit example code and run it with `sudo python2 Servo_VolumeControl.py`. The servo should rotate to the right when you press the up arrow and rotate to the left when you press the down arrow.



Left Servos are often used in robots because the joints need to articulate very precisely, and in a limited fashion

"The Adafruit board is required because a servo requires very accurate pulse-width modulation signals to set its position"

The Code SERVOS

```
import curses
from Adafruit_PWM_Servo_Driver import PWM

# Initialise the PWM device using the default address
pwm = PWM(0x40, debug=False)

servoMin = 150  # Min pulse length out of 4096
servoMax = 600  # Max pulse length out of 4096
maxDegree = 60 # Degrees your servo can rotate
degIncrease = 2 # Number of degrees to increase by each time

pwm.setPWMFreq(60) # Set PWM frequency to 60Hz

def setDegree(channel, d):
  degreePulse = servoMin
  degreePulse += int((servoMax - servoMin) / maxDegree) * d
  pwm.setPWM(channel, 0, degreePulse)

# Set up curses for arrow input
scr = curses.initscr()
curses.cbreak()
scr.keypad(1)
scr.addstr(0, 0, "Servo Volume Control")
scr.addstr(1, 0, "UP to increase volume")
scr.addstr(2, 0, "DOWN to decrease volume")
scr.addstr(3, 0, "q to quit")
scr.refresh()

degree = 60 # Start off at lowest volume
setDegree(0, degree)
```

The Code SERVOS

```
key = ''
while key != ord('q'):
  key = scr.getch()

  if key == curses.KEY_DOWN:
    degree += degIncrease

    if degree > maxDegree:
      degree = maxDegree

    setDegree(0, degree)

  elif key == curses.KEY_UP:
    degree -= degIncrease

    if degree < 0:
      degree = 0

    setDegree(0, degree)

curses.endwin()
```

"The servo should rotate to the right when you press the up arrow and rotate to the left when you press the down arrow"

Science on a Raspberry Pi

This month, learn how to do some scientific number-crunching on your Raspberry Pi

"We will look at the science you can do with these powerful little machines"

A few issues ago, we looked at using MPI with Python on your Raspberry Pi. This is useful when you are dealing with large problems that you want to spread across multiple machines. But what kind of work can you do on each of these machines? If you want to create a cluster of Raspberry Pis to do scientific work, you will want to look into using scipy to handle the actual computation tasks. Scipy is built on top of numpy, a Python module that provides enhanced number handling capabilities. Numpy uses optimised shared libraries, like LAPACK or BLAS, to handle linear algebra

type operations at the same speed as code written in C or FORTRAN. With these modules, we will take a quick look at what kind of science you can do with these powerful little machines.

Almost every Linux distribution, including the standard ones for the Raspberry Pi, include packages for scipy and numpy. To install scipy, you would execute:

```
sudo apt-get install python-scipy
```

Since numpy is part of the scipy environment, it will get automatically installed when you install scipy. In order to use them you will need to import them, usually with an alias...

```
import numpy as np
import scipy as sp
```

This import of the main scipy module only provides some core functionality that is available. All of the rest of the functions are subdivided into a number of sub-modules, such as cluster (providing clustering algorithms), integrate (providing integration and ordinary differential equation solvers), signal (for signal processing) and weave (for C/C++ integration). Each of these sub-modules needs to be imported individually. In this way, you can just import the sections you need to solve the problem at hand without cluttering up the name-space with unneeded objects.

Before we dive into scipy, it is worth taking a look at numpy and why it is worth considering Python for scientific computing. The default extended object in Python is a list. The problem with this is that a list could contain any type of object. Numpy provides an array object that can only contain elements of a single

type. With this concession, you can start to take some shortcuts because you have some meta-knowledge about the data. In regular Python, when you scale a vector you might use a for loop like this:

```
a = [1,2,3,4]
b = []
scale = 2
for curr_a in a:
  b.append(curr_a * scale)
```

In this small example, there is a lot of overhead involved. On each iteration around the for loop, Python needs to check the type of scale and the type of the current element from the list to see what it needs to do to apply the multiplication operation. When you use numpy arrays, all of this goes away. The equivalent code would look like:

```
a = np.array([1, 2, 3, 4])
scale = 2
b = scale * a
```

In this case, Python only needs to check the type of scale and a once. It then hands the two objects out to the shared linear algebra library to handle the actual multiplication step. Then the answer object comes back and is stored in b. So you get two speed-ups, one from removing all of the type checking that Python does and another from using optimised routines in the shared library.

Getting back to scipy, we should look at what we can do. A common task in scientific problem solving is

"You get two speed-ups, one from removing all of the type checking that Python does and another from using optimised routines in the shared library"

integrating a function over some range. The usual scipy function to do this is 'integrate.quad()'. You can either hand in a predefined Python function or instead create a lambda expression on the fly. As an example, say you want to integrate the sine function from 0 to 1. You could do so with:

```
import scipy.integrate as spi
import math as m
  result = spi.quad(lambda x: m.sin(x), 0.0, 1.0)
```

Hopefully, you should then be able to get the result (0.45969769413186023, 5.103669643922839e-15). The first value is the integral, while the second value is an estimate of the absolute error in the result.

Another common task is to try and fit a function to some experimental data. The usual method is to do a least-square fitting to minimise the difference between the function and the experimental data. The function 'leastsq' is in the 'scipy.optimize' sub-module. To do this type of calculation, you need to provide a function to compute the residuals, and a starting point for each of the adjustable parameters in this function. As an example, say you had some measurements that seemed to be in a sinusoidal shape. You could try fitting a general sinusoidal function:

```
y = A * sin(2*pi*k*x + theta)
```

...to these measurements. The mathematical residual function would be:

```
residuals = y - A*sin(2*pi*k*x + theta)
```

…and the adjustable parameters would be 'A', 'k' and 'theta'. In order to use the 'leastsq' Python function, you would need to write this residual equation as a Python function that does the relevant calculation. Then you can hand in the function, an array with an initial guess for the parameters, plus any arguments needed for the residual function to the function 'optimize.leastsq()'.

The last example we are going look at is solving linear systems of equations. This is something done by many disciplines, such as physics and engineering. You can do this by using the linear algebra module (linalg). Using matrix notation, the system is defined by 'Ax=b', where 'A' is the matrix of coefficients, 'x' is the vector of variables and 'b' is the vector containing the right-hand sides of each equation. If you have an 'A' and 'b' defined, you can solve this problem just by using:

```
import scipy.linalg as sla
x = sla.solve(A,b)
```

Since it uses the new data-types provided by numpy, it simplifies the code you need to write, making it easier to understand later on what you were trying to write.

Hopefully we have covered enough in this article to convince you to give scipy and numpy a try with any problems you may need to work on. There are hundreds of functions available to do rather complicated calculations – check out **http://docs.scipy.org/doc/numpy/reference**. And you can't beat the power usage per flop that you can get from the Raspberry Pi.

"Since it uses the new data-types provided by numpy, it simplifies the code you need to write, making it easier to understand later on what you were trying to write"

The Code SCIENCE ON PI

```
# The start of all scientific code should
# import numpy and scipy
import numpy as np
import scipy as sp
```

```
# The old way of multiplying a vector
# by a scale factor is by using
# a for loop
a = [1, 2, 3, 4]
b = []
scale = 2
for curr_a in a:
    b.append(curr_a * scale)
```

```
# The cleaner, optimized way is by
# using the numpy array data-type
a = np.array([1, 2, 3, 4])
scale = 2
b = scale * a
```

```
# To integrate sin(x) over 0 to 1,
# we need to import the integrate
# sub-module and use the quad function
import scipy.integrate as spi
import math as m
# Result will contain both the answer
# and an estimate of the error in the
# answer
result = spi.quad(lambda x: m.sin(x), 0.0, 1.0)
```

The Code SCIENCE ON PI

```
# This leastsq example is from
# the scipy docs
x = np.arange(0, 6e-2, 6e-2/30)
A, k, theta = 10, 1.0/3e-2, pi/6
# For testing, we will add noise to the true
# values for y
y_true = A * np.sin(2*pi*k*x + theta)
y_meas = y_true + 2*np.random.randn(len(x))
def residuals(p, y, x):
    A, k, theta = p
    err = y - A*np.sin(2*pi*k*x + theta)
    return err
p0 = [8, 1/2.3e-2, pi/3]
from scipy.optimize import leastsq
plsq = leastsq(residuals, p0, args=(y_meas, x))
```

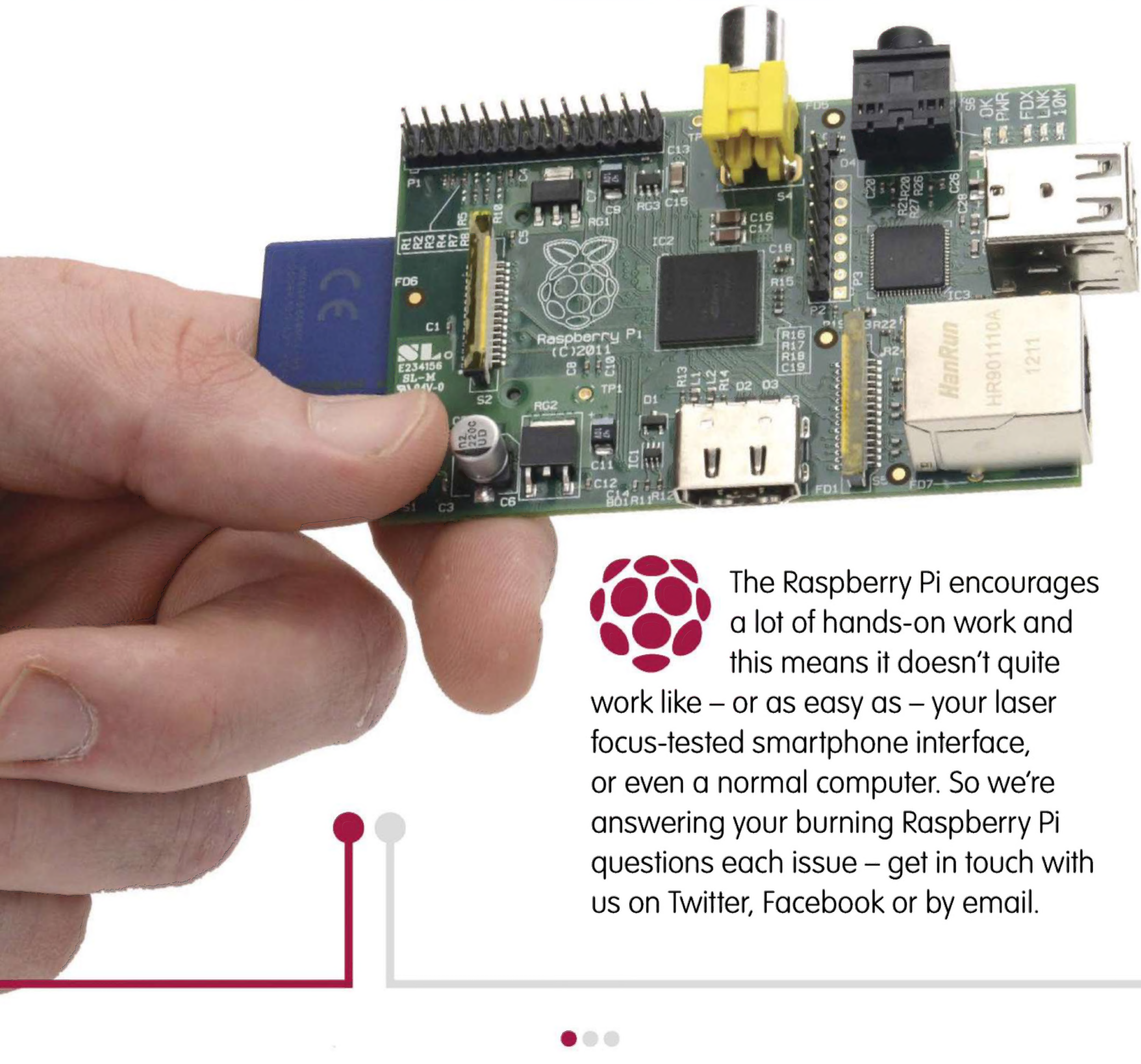
```
# To solve systems of equations
# need to import the linalg sub-module
import scipy.linalg as spl
# A will be the matrix of coefficients
A = np.array([[1,2],[3,4]])
# b will be the vector of right-hand
# side values
b = np.array([[5],[6]])
# We can find the values of x with '€˜solve'€™
x = spl.solve(a, b)
```

"To use the 'leastsq' Python function, you would need to write this residual equation as a Python function that does the relevant calculation"

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

Do I need to get a microSD card reader for my computer before getting a new Pi?
Laura via Facebook

If you already have an SD card reader in your PC or laptop, you should be fine because most microSD cards come with an SD card adapter. Double-check which one you're buying though to make sure it does, and be prepared to replace it as some of the adapters seem to break after not too long. The change is worth it, as the new microSD cards really reduce the physical footprint of the Raspberry Pi.

Can I use my Raspberry Pi as a portable mp3 player?
Tim L. via email

Technically there's no reason why you couldn't turn a Raspberry Pi into an mp3 player – you'd need a case, a portable battery and you can stick headphones into the 3.5 mm headphone jack. You'd have to wire up physical buttons to control it though, or figure out a remote control method via your phone… which you could probably listen to music on anyway. You could also try adding the PiTFT mini touch screen in order to get track information and create a custom interface for that.

Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

Ben Nuttall: Sous vide eggs I cooked yesterday with the updated #chefhat library for Raspberry Pi. 2 hours at 63C **bit.ly/1FBoPHO**

PiBorg: Little @Raspberry_Pi #robot #DiddyBorg can see, by adding our ultrasonic mount kit! Awww. **piborg.org/accessories**

PiJuice: .@kickstarter UPDATE: PiJuice Maker Kits for the @Raspberry_Pi now available - get 'em while they're hot!! **https://kck.st/18liEoy**

Raspberry Pi: Picademy is coming to THE NORTH! Free CPD for teachers at the @NtlSTEMCentre in York in May **bit.ly/1MKk7tx**

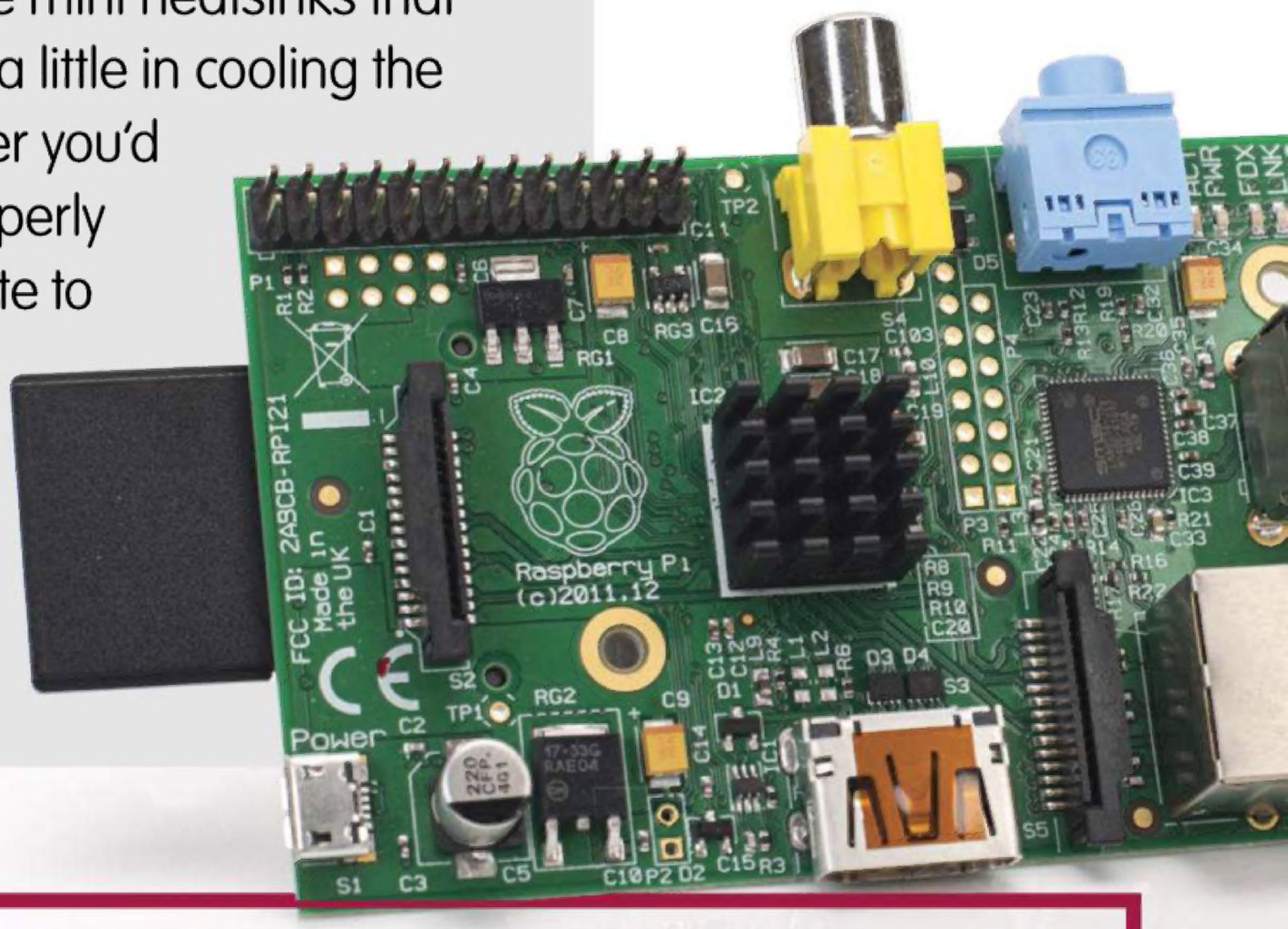
>>

What do I need to start soldering?
Dev via Twitter

Soldering really requires two things: a soldering iron and some solder. The solder itself is a conductive metal sold in a sort of wire-like shape that melts when you heat it up on the tip of a soldering iron. There are various others tools you can get, such as a pen that will suck up excess solder that hasn't quite cooled. The main thing to remember is to solder on a fire retardant surface and to be careful not to burn yourself.

Do I need to get a heat sink for my Raspberry Pi?
Tab K via Facebook

The Raspberry Pi can get warm if used in certain ways, especially if you're overclocking the processor, however the air cooling on the Pi is generally good enough to keep it from breaking. There are some mini heatsinks that exist which can help a little in cooling the Raspberry Pi, however you'd need to affix one properly and use thermal paste to make sure the heat transfers properly, otherwise you won't get much out of it.



Raspberry Pi: #PiDay from sea to shining sea (and beyond!) **bit.ly/1AfNdcc**

Raspberry Pi: New post! Tech specs of Astro Pi, our space hardware launching later this year. It could be running YOUR experiment! **bit.ly/1BQdjWc**

Raspberry Pi: Raspberry Pi celebrates third birthday as sales reach five million **bit.ly/1EwagUn** ...via @INQ

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

AMAZING
PI PROJECTS

- Retro arcade
- Weather station
- Web radio **& more**

Get this issue's source code at:
www.linuxuser.co.uk/raspicode